UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Controlo de Acesso no Sistema Académico Fenix

Daniel Filipe Mendes Pires

Mestrado em Informática

Trabalho de projeto orientado por:
Prof. Doutora Maria Dulce Pedroso Domingos
Prof. Doutor Carlos Nuno da Cruz Ribeiro

2019

“java.lang.NullPointerException”

– Java

Agradecimentos

Um simples obrigado não chega para agradecer a todos aqueles que fizeram parte desta minha jornada do Mestrado em Informática.

Antes de mais quero agradecer à Reitoria da Universidade de Lisboa por me ter proporcionado esta oportunidade de desenvolver a minha tese de mestrado.

Aos meus orientadores Professora Dulce Domingues e Professor Carlos Ribeiro.

À equipa de desenvolvimento do Fenix do Instituto Superior Técnico e à empresa Quorum Born IT pela colaboração prestada.

A todos os colegas do Departamento de Informática, especialmente ao Núcleo de Desenvolvimento de Software, nomeadamente ao Ricardo Rito, Pedro Moita, Nuno Heitor, Catarina Silva, Daniela Mendes, Fábio Ferreira e Daniel Vitoriano. Aos colegas do Núcleo de Gestão de Sistemas de Informação, Ana Rute, José Lima, Andreia Rainha, André Faria, Teresa Gouveia, Tânia Castelo e Tânia Crespo.

Aos colegas bolseiros que entraram ao mesmo tempo que eu nesta nova aventura e nos sempre demos muito bem ao longo deste caminho, Ricardo Carvalho, César Mendes, Ana Espinheira, João Batista e Nuno Mendes.

À minha família que sempre me apoiou e me guiou no caminho para o sucesso. Muito obrigado pai, mãe e irmã.

Por ultimo mas não menos importante e com um papel fulcral na minha sanidade mental, quero agradecer à minha namorada Margarida Anastácio.

Espero não me ter esquecido de agradecer a alguém. A todos um grande obrigado.

Resumo

O Fenix é um projeto universitário que surgiu em 2002 no Instituto Superior Técnico e começou por ser um sistema de informação interno, com o objetivo de facilitar a gestão académica. Atualmente está implementado em dezasseis escolas da Universidade de Lisboa.

Com o aumento do uso do sistema académico Fenix pelas dezasseis escolas, há cada vez mais a necessidade da reorganização do sistema de controlo de acesso, uma vez que este tem várias funcionalidades com acesso restrito. O sistema atual tem processos pouco ágeis para a gestão de permissões devido a restrições a nível de interface e de implementação do sistema de controlo de acesso.

Com este trabalho pretende-se que seja construído um sistema de controlo de acesso que facilite a gestão de permissões e a torne mais ágil. Com base na análise dos requisitos identificados foi criada uma solução que implementa um sistema de controlo de acesso baseado em perfis. Adicionalmente foram criadas interfaces que permitem corresponder às necessidades dos utilizadores e do novo sistema de controlo de acesso.

Palavras-chave: Fenix, Controlo de Acesso, Permissões, RGPD, Escalabilidade, Flexibilidade, Perfis

Abstract

Fenix is an academic project that emerged in 2002 at Instituto Superior Técnico and started as an internal information system, with the objective of facilitating academic management. It is currently implemented in sixteen schools of the University of Lisbon.

With the increase in the use of the Fenix academic system by the sixteen schools, there is an increasing need for the reorganization of the access control system, since it has several functionalities with restricted access. The current system has weak processes for managing permissions due to interface and implementation restrictions of the access control system.

This work intends to build an access control system that facilitates the management of permissions and makes it more agile. Based on the analysis of the identified requirements, we have created a solution that implements a role-based access control system. In addition, interfaces have been created to meet the needs of users and the new access control system.

Keywords: Fenix, Access Control, Permissions, RGPD, Scalability, Flexiblity, Roles

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O Fenix é uma plataforma de software modular para gestão académica e administrativa de instituições de ensino superior [6]. O Fenix encontra-se implementado em dezasseis escolas da Universidade de Lisboa, incluindo os serviços centrais da Reitoria. Um dos principais benefícios decorrentes da implementação do Fenix advém da capacidade de personalização modular do software consoante as necessidades das escolas.

1.1 Problema

O Fenix nutre de inúmeras funcionalidades com acesso restrito, onde é necessário a existência de mecanismos para gestão de permissões ou grupos de acesso a páginas e opções de menus, para os utilizadores.

A necessidade de mecanismos de gestão de permissões de utilizadores, foi desde muito cedo identificada. No entanto, com o sistema atual do Fenix, as operações de atribuição e remoção de permissões consistem em processos pouco ágeis e complexos, o que dificulta as tarefas de gestão de acessos. Com o uso do Fenix em diferentes escolas, bem como com o constante aumento de utilizadores no sistema académico, foi identificada a necessidade de reequacionar os mecanismos de gestão e colmatar lacunas existentes no sistema, tanto de usabilidade como de implentação.

A 25 de Maio de 2018, a entrada em vigor do Regulamento Geral de Proteção de Dados (RGPD) veio criar legalmente a implicação de resposta a questões de auditoria, por parte de sistemas de informação existentes. Com este trabalho pretende-se identificar se o Fenix se encontra preparado para responder aos requisitos exigidos pelo RGPD, do ponto de vista de controlo de acesso e de auditoria.

1.2 Abordagem

Inicialmente, foi efetuado o levantamento e a análise de requisitos, que permitiu identificar as necessidades dos utilizadores perante o sistema. Com este trabalho efetuado,

foi possível construir uma base de conhecimento com as principais lacunas que o sistema apresentava, e criar uma solução para resolver as problemáticas identificadas.

Foi possível criar uma solução baseado em conceitos base de controlo de acesso aplicados à realidade da gestão do sistema académico, que posteriormente foi validada pelas equipas técnicas dos serviços centrais e das escolas que utilizam a ferramenta de gestão de permissões. Nas sessões realizadas, para além da apresentação do protótipo desenvolvido, foi possível identificar oportunidades de melhoria e erros.

1.3 Contribuição

De forma a responder aos requisitos identificados, foi proposto um modelo de controlo de acesso baseado em perfis para o sistema Fenix. Com este modelo é possível, por exemplo, usufruir dos mecanismos de composição de perfis e herança de permissões.

Adicionalmente, este modelo foi integrado com o sistema de controlo de acesso atualmente implementado no Fenix, já que foi identificado que não era possível efetuar a sua substituição de forma imediata porque o código que concretiza o controlo de acesso está, em parte, disseminado pelo sistema.

A implementação deste modelo de controlo de acesso inclui a criação de interfaces para corresponder às necessidades dos utilizadores. A validação dos critérios de usabilidade destas interfaces foi efetuada recorrendo ao método *System Usability Scale* [17].

1.4 Estrutura do documento

Este documento segue a seguinte estrutura:

- Capítulo 2 - Contexto

  Descreve o contexto do trabalho e define conceitos transversais sobre controlo de acessos, bem como a arquitetura atual de gestão de acessos do sistema integrado de gestão académica Fenix.

- Capítulo 3 - Controlo de acesso baseado em perfis para o Fenix

  Neste capítulo é descrita a análise de requisitos, o desenho e a implementação do mesmo.

- Capítulo 4 - Avaliação experimental

  Descreve a forma como foi avaliada a solução, é feita a apresentação e análise dos dados.

- Capítulo 5 - Conclusões e trabalho futuro

  Conclusões do trabalho efetuado e descrição do trabalho para o futuro.

Capítulo 2

Contexto

Neste capítulo é apresentado o sistema integrado de gestão académica Fenix, dando particular detalhe aos aspetos sobre controlo de acesso. Previamente são sistematizados os conceitos relacionados com controlo de acesso.

2.1 Controlo de acesso

Um sistema de controlo de acesso permite mediar a interação entre o utilizador e os recursos de um sistema. Este implementa políticas de segurança que definem quem acede, como acede, quando acede e ao que acede [27, Chapter 4.1]. Existem vários tipos de políticas que permitem definir um conjunto diferente de regras. Um sistema pode adotar uma ou várias políticas de controlo de acesso.

Nesta secção são apresentados os conceitos base de controlo de acesso e as principais políticas de controlo de acesso.

2.1.1 Conceitos base

Na figura 2.1 está representado um sistema de controlo de acesso, onde existem três funções principais:

- **Autenticação**: Verifica que as credencias de um utilizador são válidas [27, Chapter 4.1].
- **Autorização**: Verifica se o utilizador tem acesso aos recursos de um sistema [27, Chapter 4.1].
- **Auditoria**: Monitoriza e armazena o histórico da atividade de um sistema de forma a verificar a conformidade com as suas políticas de acesso, averiguando se houve falhas de segurança [27, Chapter 4.1].

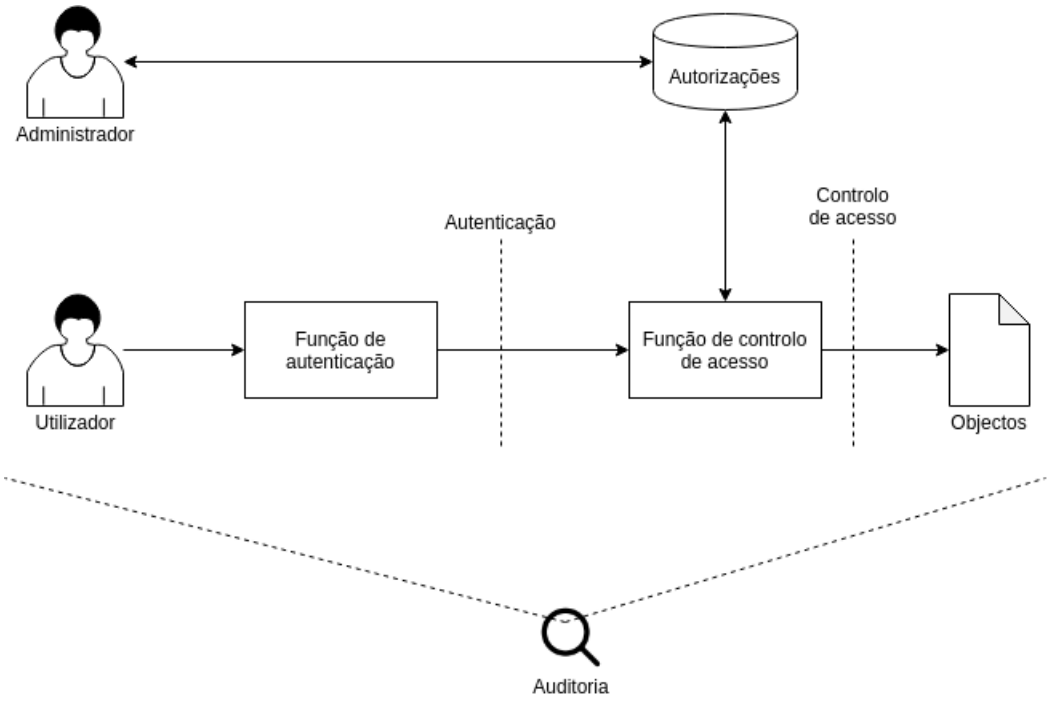

Figura 2.1: Relação entre controlo de acesso e funções de segurança

Os conceitos base de controlo de acesso são sujeito, objecto e direitos de acesso. Os conceitos adicionais serão definidos à medida que forem sendo introduzidos.

Sujeito

O sujeito é a entidade capaz de aceder a objetos. Pode ser um utilizador, um processo ou as próprias máquinas [27, Chapter 4.2].

Objeto

Um objeto é um recurso com acesso controlado pelo sistema. Pode, por exemplo, ser um ficheiro com informação, programas, entre outros [27, Chapter 4.2].

Direitos de acesso

Os direitos de acesso definem a forma como um sujeito pode aceder a um determinado objeto [27, Chapter 4.2]. Os direitos de acesso, definidos para um determinado tipo de objecto, dependem das suas características. Por exemplo, os sistemas de ficheiros suportam os direitos de acesso de escrita, leitura e execução para os seus objectos, ficheiros e directorias. Os sistemas de gestão de bases de dados definem direitos de acesso de *insert*, *select*, *create table*, etc.

	Objecto 1	Objecto 2	Objecto 3	Objecto 4
Utilizador A	Proprietário Ler Escrever		Proprietário Ler Escrever	
Utilizador B	Ler	Proprietário Ler Escrever	Escrever	Ler
Utilizador C	Ler Escrever	Ler		Proprietário Ler Escrever

Figura 2.2: Matriz de acessos

2.1.2 Controlo de acesso discricionário

O controlo de acesso discricionário é baseado na identidade do utilizador que pede acesso e nas regras de acesso (designadas por permissões) que definem o que o utilizador está ou não autorizado a fazer [27, Chapter 4.3]. Este tipo de controlo de acesso permite que um utilizador com acesso a um objeto, normalmente o dono do objeto, autorize outro utilizador a ter acesso ao mesmo objeto.

A figura 2.2 ilustra uma matriz de acesso, onde por exemplo o utilizador C tem sobre o objecto 1 os direitos de acesso de ler e escrever. Na prática, esta matriz pode ser implementada através de listas de controlo de acesso (*ACL*) ou através de listas de capacidades, conforme é detalhado de seguida.

Lista de controlo de acesso

As ACLs apresentam, para cada objecto, os direitos de acesso de cada utilizador ou grupos de utilizadores, como ilustrado na figura 2.3, e podem conter entradas com direitos acesso por omissão de forma que utilizadores não especificados possam usufruir desses direitos [27, Chapter 4.3].

Esta visão sobre a matriz de acessos permite determinar os utilizadores que tem acesso a um determinado objecto, mas não é conveniente se for necessário determinar todos os direitos de acesso de um utilizador [27, Chapter 4.3].

Lista de capacidades

Ao contrário da lista de controlo de acesso, a lista de capacidades, como representado na figura 2.4, especifica os direitos de acesso por utilizador [27, Chapter 4.3].

As vantagens da lista de capacidades são opostas às vantagens das ACLs. Desta forma, é possível obter os direitos de acesso de cada um dos utilizadores, mas é mais complexo obter os direitos de acesso dos utilizadores para um determinado objecto [27, Chapter 4.3].

Um exemplo de aplicação do controlo de acesso discricionário é o sistema de ficheiros do sistema *Unix*. Neste sistema estão previstos três tipos de sujeitos:

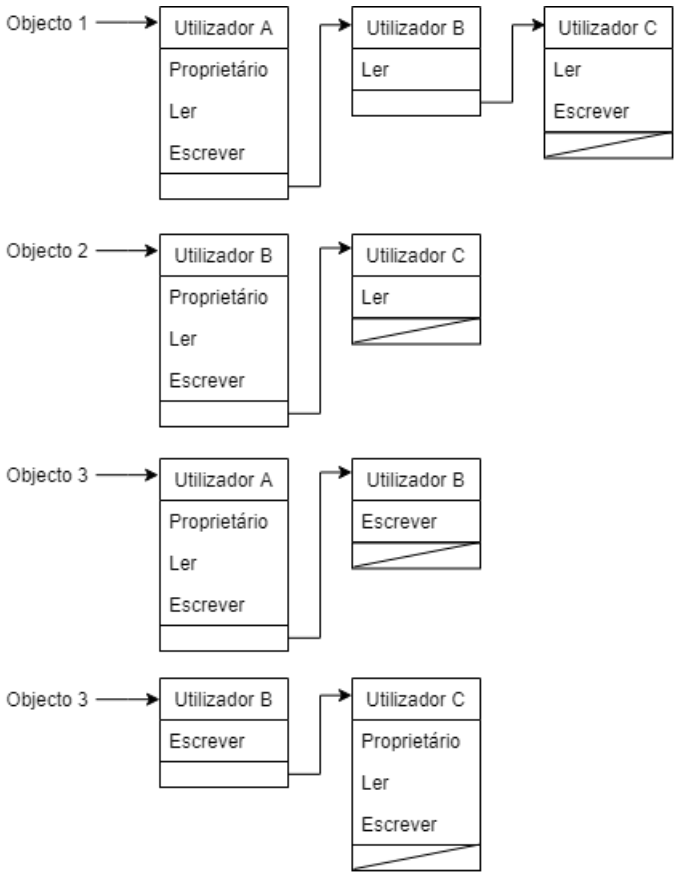

Figura 2.3: Lista de controlo de acesso

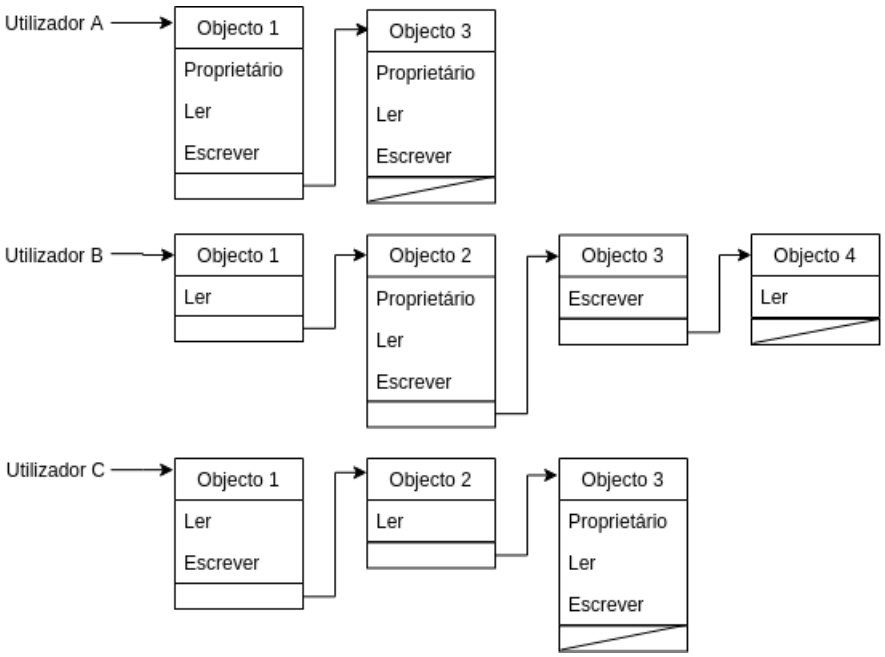

Figura 2.4: Lista de capacidades

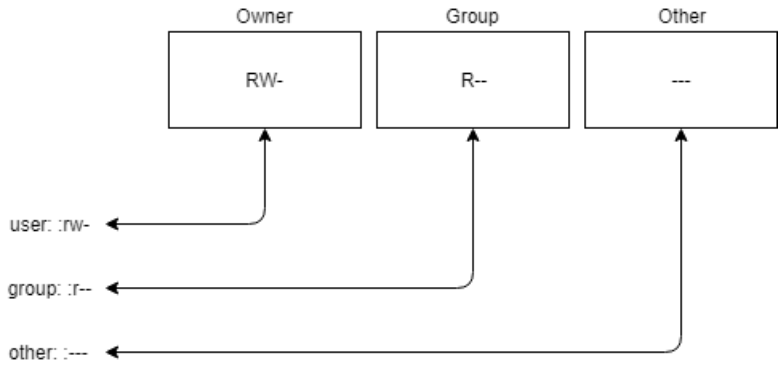

Figura 2.5: Exemplo de controlo de acesso discricionário

- Dono

  O dono é, regra geral, o criador de um objeto. O dono pode definir os direitos de acesso ao ficheiro dos restantes sujeitos.

- Grupos

  Os grupos são constituídos por utilizadores. Os ficheiros e directorias estão associados a um grupo de utilizadores. Os utilizadores pertencentes a um determinado grupo podem usufruir das permissões definidas para esse grupo.

- Outros ou Mundo

  Os restantes utilizadores que não fazem parte do grupo associado ao ficheiro.

A cada um destes sujeitos podem ser atribuídos direitos de acesso de leitura, escrita e/ou execução, conforme ilustrado pela figura 2.5.

2.1.3 Controlo de acesso mandatório

No controlo de acesso mandatório são associados níveis de segurança aos sujeitos e aos objetos [27, Chapter 4.1]. Quando associados a um objecto, os níveis de segurança indicam o seu grau de secretismo.

Um exemplo deste tipo de controlo de acesso é o modelo *Bell-LaPadula*, em que um utilizador só consegue ler dados com o nível de segurança igual ou abaixo do seu e só consegue escrever dados com o nível de segurança igual ou superior ao seu [15, Chapter 2.1.1]. O modelo *Bell-LaPadula* permite manter a confidencialidade dos dados.

Neste tipo de controlo de acesso, os donos dos objectos não podem administrar as suas permissões.

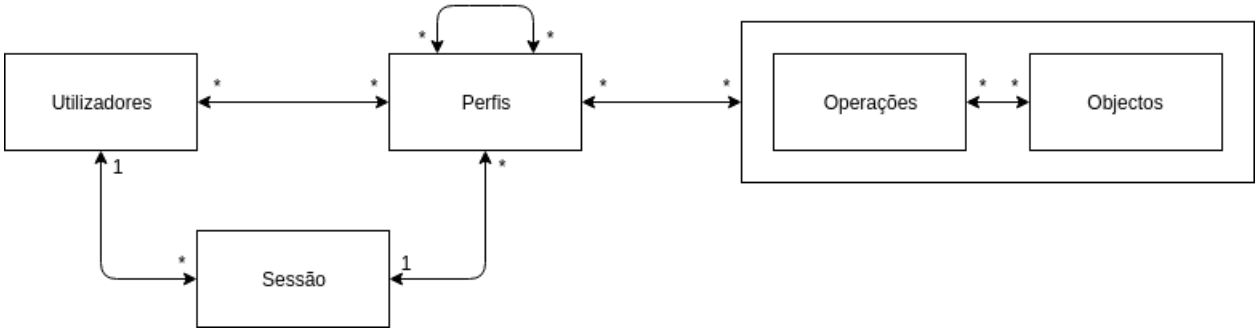

Figura 2.6: Modelo base de controlo de acesso baseado em perfis

2.1.4 Controlo de acesso baseado em perfis

O conceito base do controlo de acesso baseado em perfis, do inglês *role-based access control*, também traduzido para controlo de acesso baseado em papéis, é o perfil [27, Chapter 4.5]. Um perfil está associado à estrutura orgânica da organização ou às funções desempenhadas pelos utilizadores. Os utilizadores são associados a perfis e as permissões são atribuídas aos perfis. Os utilizadores podem usufruir das permissões dos perfis aos quais estão associados [27, Chapter 4.5].

Na figura 2.6 podemos ver o esquema do modelo de controlo de acesso baseado em perfis. Um utilizador pode ter mais do que um perfil num determinado sistema, a diferenciação do perfil a ser usado depende da sessão atual do utilizador no sistema.

A definição do controlo de acesso baseado em perfis inclui a especificação de um conjunto de modelos: o modelo base, o modelo hierárquico e o modelo de restrições [26].

Modelo base

Este modelo contém as funcionalidades mínimas do controlo de acesso baseado em perfis, não havendo qualquer hierarquização ou restrições.

Modelo hierárquico

Tipicamente, uma organização está estruturada de acordo com uma hierarquia de perfis. Este modelo suporta esta hierarquia permitindo que os perfis herdem, implicitamente, as permissões associadas a perfis subordinados [27, Chapter 4.5]. De acordo com o exemplo da figura 2.7, os utilizadores com o perfil “Líder de projecto 1” herdam as permissões dos perfis de “Engenheiro de produto 1” e de “Engenheiro de Qualidade 1”, os quais herdam as permissões do perfil “Engenheiro 1” e assim sucessivamente [27, Chapter 4.5].

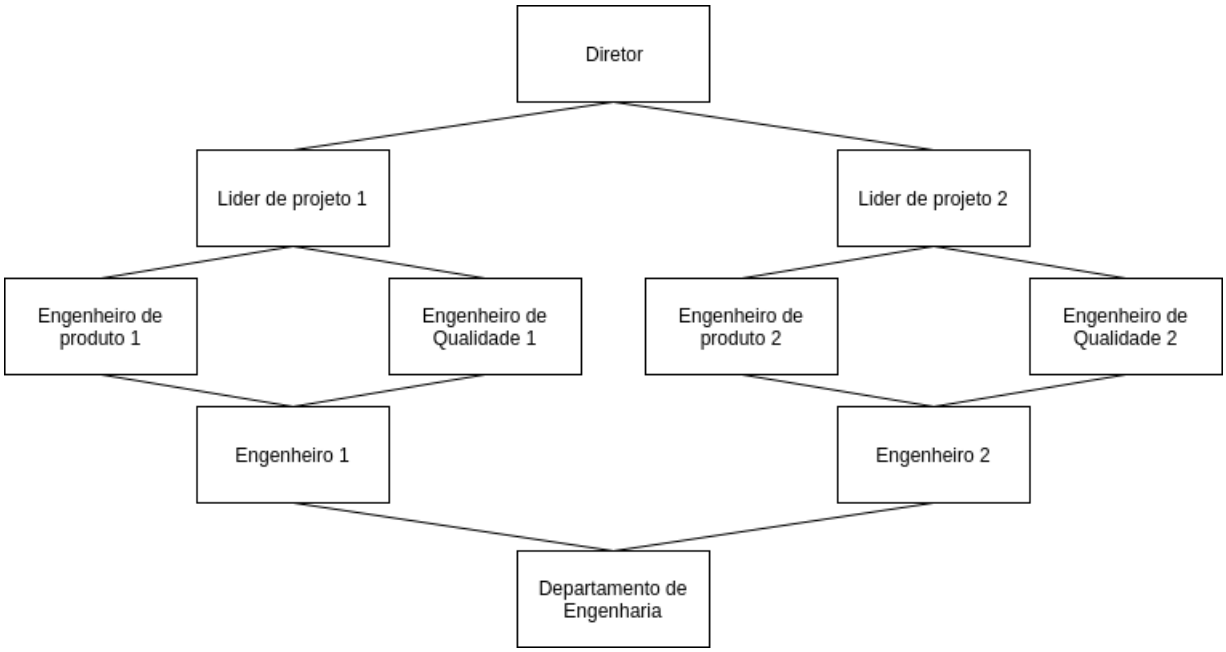

Figura 2.7: Modelo de hierarquia de perfis (adaptado de [27])

Modelo de restrições

O modelo de restrições, apresentado na figura 2.8, parte do modelo base e permite adaptar-se à especificidade das políticas de segurança da organização. As restrições definem as regras de atribuição e utilização dos perfis. Pode-se, por exemplo, definir perfis mutuamente exclusivos, a cardinalidade de utilizadores por perfil ou de permissões por perfil, e pré-requisitos para determinados perfis [26], conforme detalhado de seguida:

- Perfis mutuamente exclusivos são perfis que não podem ser atribuídos em simultâneo a um determinado utilizador, quando considerado o seu comportamento estático. No comportamento dinâmico, um utilizador não pode ter perfis mutuamente exclusivos em simultâneo na mesma sessão [26].
- Cardinalidade refere-se ao número máximo de utilizadores para determinado perfil, restringindo o número de utilizadores que este pode ter [26]. Por exemplo, um perfil de chefia de um departamento só deve ser atribuído a um utilizador. Também pode ser a restrição do número de permissões atribuídas a um perfil [26].
- Pré-requisitos permitem definir, por exemplo, que um utilizador só pode ser associado a um perfil se já tiver sido associado a outro perfil [26].

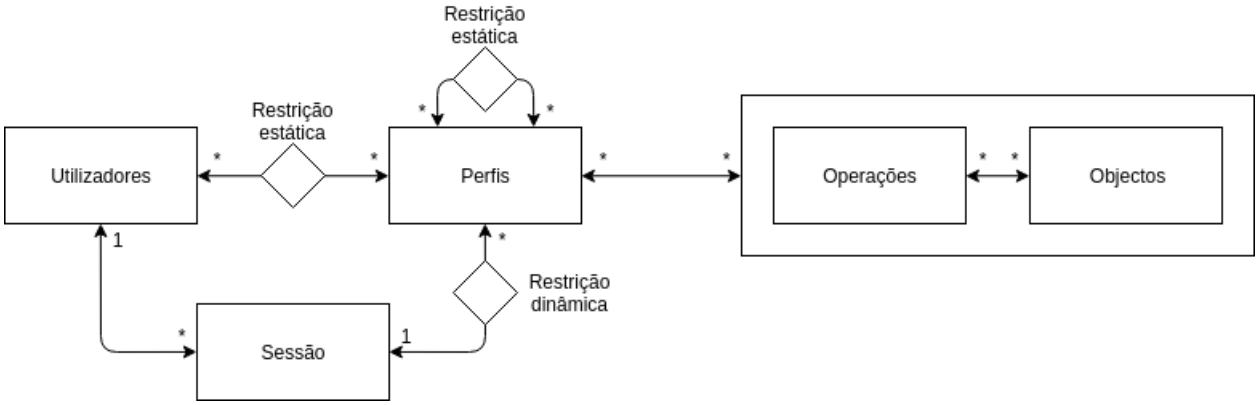

Figura 2.8: Modelo de controlo de acesso baseado em perfis com restrições

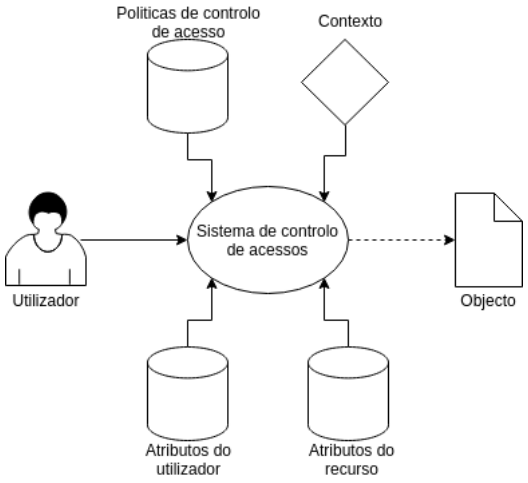

Figura 2.9: Controlo de acesso baseado em atributos

2.1.5 Controlo de acesso baseado em atributos

O controlo de acesso baseado em atributos permite associar restrições (ou condições) às permissões. As permissões são definidas através de um tuplo (sujeito, objecto, direitos de acesso, condição). Um sujeito só pode usufruir do direito de acesso, a um determinado objecto, definido numa permissão se a respectiva condição for verdadeira. As condições podem incluir informação sobre os atributos do sujeito e do objecto a ser acedido, bem como do contexto do ambiente. [27, Chapter 4.6]

Este tipo de controlo de acesso permite uma maior expressividade na definição das permissões, mas diminui o desempenho na verificação dos acessos, uma vez que é necessário avaliar as condições, obtendo os atributos utilizados na sua definição, como ilustrado na figura 2.9.

2.2 Sistema integrado de gestão académica (SIGA) Fenix

O sistema Fenix é um projeto universitário que surgiu em 2002 no Instituto Superior Técnico e começou por ser um sistema de informação interno, com o objetivo de facilitar a gestão académica. Em 2015, a plataforma de gestão académica foi instalada, num projeto transversal da Universidade de Lisboa, na maioria das escolas. Os três principais desafios lançados com este projeto foi a uniformização dos sistemas académicos das diferentes escolas, criação de uma comunidade de desenvolvimento para aquisição de conhecimento interno e redução de custos de manutenção.

Esta secção descreve o sistema Fenix, enumerando as principais tecnologias utilizadas, a arquitetura do sistema e a implementação do controlo de acesso.

2.2.1 Tecnologias

Atualmente o sistema utiliza várias tecnologias:

- Java 8

  O Java é uma linguagem de programação orientada a objetos. É compilada para *bytecode* o qual é interpretado por uma máquina virtual, a *Java Virtual Machine*. Deste modo, o código gerado é independente da plataforma onde será executado. Atualmente é usado numa grande variedade de ambientes e plataformas.

- Java Server Pages

  Java Server Pages é uma tecnologia que permite criar de forma simples e rápida, páginas *web* dinâmicas baseadas em *HTML* e *XML* [11].

  Um dos grandes benefícios desta ferramenta é a integração com as *APIs* Java, como por exemplo *JDBC*, *EJB*, *JAXP* [11].

  É uma solução de interface que apresenta vantagens relativas ao desempenho, pois incorpora elementos dinâmicos na página e é compilado antes de ser processado pelo servidor, em vez de ser compilado em tempo de execução. Por este motivo apresenta maiores níveis de eficiência [11].

- Struts

  Struts é uma framework de desenvolvimento baseada no modelo *Model-View-Controller*, para aplicações *web* [22]. Esta ferramenta, de desenvolvimento modular com base na linguagem Java, disponibiliza *APIs* de integração com a camada de interface e base de dados [14].

- Spring

  A framework Spring é uma framework de desenvolvimento de software em Java baseada no modelo *Model-View-Controller*, e é utilizado num *web container*. A framework Spring é dividida em módulos, permitindo a gestão de dependências para um projeto, consoante as necessidades, tais como, por exemplo a injeção de dependências, transações e persistência de dados, camada web, entre outros [12].

- Maven

  Maven é uma ferramenta que permite automatizar o processo de gestão de dependências dos projectos, utilizando para isso um ficheiro no formato *XML* denominado de *Project Object Model*. Desta forma, é possível simplificar e normalizar o processo de construção de projetos Java, com a gestão remota de dependências, acedendo a diferentes repositórios [9].

- MySQL

  MySQL é um sistema *open-source* para gestão de base de dados relacionais [10].

- Fenix Framework

  Suporta o desenvolvimento de aplicações baseadas em Java que necessitem de um modelo de domínio, permitindo que o programador se abstraia da camada de persistência.

  O domínio da aplicação é definido através de uma linguagem criada especificamente, designada por *Domain Modeling Language (DML)* [19].

  A *framework* Fenix gera automaticamente a camada responsável pela persistência dos dados, fazendo a associação entre os objectos de domínio e a base de dados, assegurando ainda a sua sincronização e versionamento. Na secção 2.2.2 são apresentados mais detalhes sobre esta *framework*.

  Apesar da grande vantagem de abstrair o programador da gestão da camada de persistência, apresenta limitações para o programador, tais como a falta de controlo da correspondência dos objetos para a base de dados.

- Bennu

  Bennu é a base para construir aplicações *web* modulares em Java, baseado na *framework* Fenix. Contém *APIs* que cobrem funcionalidades como a autenticação de utilizadores, grupos de acesso, transações de escrita, indexação e pesquisa de informação. O Bennu disponibiliza também mecanismos para configuração de instalações em execução no ambiente Fenix e renderização de configurações de menus [13].

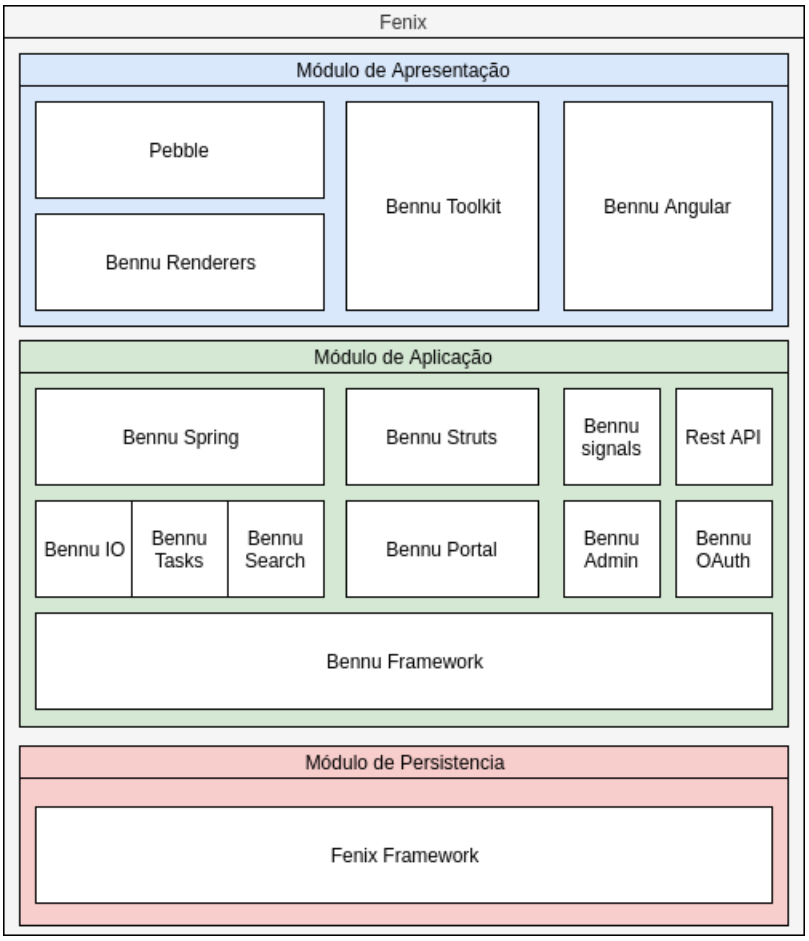

Figura 2.10: Arquitetura do Fenix

- Angular

  Angular é uma *framework JavaScript open-source* que permite a construção de interfaces *web* com recurso a *HTML*, *CSS* e principalmente *JavaScript*. Esta *framework* disponibiliza um conjunto de componentes para geração de *templates*, criação de serviços e injeção de dependências, que automatizam as tarefas e facilitam a execução de testes unitários em aplicações [1].

2.2.2 Arquitetura

A arquitetura do Fénix está organizada em três módulos logicamente separados. Na figura 2.10 estão representados os três módulos: o módulo de persistência que é responsável pelo armazenamento dos dados; o módulo de aplicação que interpreta os pedidos e o módulo de apresentação que representa a interface com o utilizador.

Esta arquitetura segue o modelo *Model-View-Controller* [22], modelo bastante usado para a construção de aplicações *web*, que permite que os módulos cooperem entre si abstraindo-se da forma como cada um deles está implementado.

Este tipo de arquitetura permite que cada um dos módulos evolua de forma independente, tornando a aplicação flexível e escalável, facilita a manutenção, pois é possível alterar por exemplo o módulo de apresentação sem se mexer em qualquer um dos outros módulos.

De seguida são detalhados os módulos que compõem a arquitetura do sistema Fenix.

Módulo de persistência

O módulo de persistência é composto pela *framework* Fenix. Utilizando esta *framework*, o modelo de domínio é especificado através da linguagem DML [19, Chapter 5]. A partir desta especificação são criadas de forma automática as classes e as suas relações.

Para assegurar a persistência dos objetos na base de dados relacional, a sua correspondência para o modelo relacional é feita utilizando a técnica de *Object Relational Mapper* [23].

Esta *framework* também permite gerir as transações feitas na base de dados e garante a coerência da mesma. Usa uma *Java Versioned Software Transactional Memory* que permite através de *Versioned Boxes* fazer com que seja possível manter histórico [18]. Cada transação obtém o número de versão do objeto para garantir que cada leitura é feita na versão mais recente no momento da transação, fazendo com que não haja conflitos de leitura. As escritas criam sempre uma versão nova do objeto em vez reescrever a versão existente [18].

Módulo de aplicação

O módulo aplicacional permite fazer a ponte entre o módulo de apresentação e o módulo de persistência. Este módulo é composto pelos módulos Bennu.

Estes módulos são responsáveis, por exemplo, pela gestão de utilizadores, menus e configuração da plataforma. Cria uma abstração da camada de persistência, o que acelera o desenvolvimento das aplicações para o programador.

Módulo de apresentação

O módulo de apresentação é a interface da aplicação *web* e define como a aplicação é vista pelo utilizador. Pode ser separada por módulos e implementada por várias ferramentas diferentes.

A forma como o sistema Fenix está arquitetado permite o recurso a diferentes ferramentas de apresentação, consoante a necessidade do projeto que se pretende implementar.

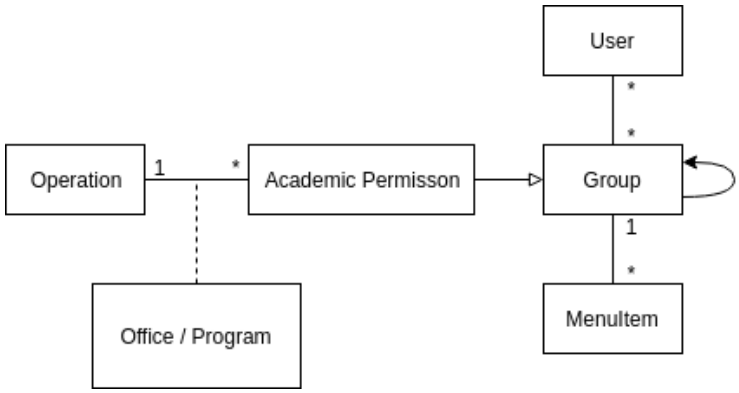

Figura 2.11: Modelo simplificado de controlo de acesso do Fénix

O módulo Bennu *Toolkit* é um dos exemplos de uma ferramenta que fornece o lado de cliente do módulo aplicacional Bennu.

2.2.3 Modelo de controlo de acesso do Fénix

No sistema integrado de gestão académica Fénix, o controlo de acesso é gerido através de grupos de acesso. Os utilizadores são associados a grupos e os grupos são associados aos menus conforme ilustrado na figura 2.11.

Na plataforma fénix existem três tipos de grupos:

- Grupos calculados pelo sistema - Estes grupos são pré-definidos no Fenix. Permitem distinguir, a nível de sistema, os vários tipos base de utilizadores, como por exemplo alunos e professores.

- Grupos *ad-hoc* - Grupos cujo nome tem o prefixo #, também designados por grupos dinâmicos no contexto da *framework* Fenix, são grupos criados por utilizadores que pertencem ao grupo administrador do sistema. É possível adicionar e remover qualquer utilizador a um dado grupo *ad-hoc*. Um exemplo deste tipo de grupo é o *#schooladmin*.

- Grupos de permissões académicas - Estes grupos dão acesso a funcionalidades que dependem de um nível mais fino de granularidade, tal como o curso ou a secretaria, conforme ilustrado na figura 2.12. As permissões académicas apenas são aplicadas no módulo académico do sistema Fenix. Um utilizador que pertença a um grupo de permissão académica para um determinado curso pode, por exemplo, listar os alunos apenas desse curso. Nos exemplos apresentados na figura 2.12, o utilizador João tem permissões para matricular alunos de todos os cursos que pertençam à secretaria "Divisão Académica", enquanto que o utilizador Bruno tem permissões para matricular alunos para os cursos listados. Apesar das permissões estarem associadas a utilizadores, de forma a manter a coerência com o modelo de controlo

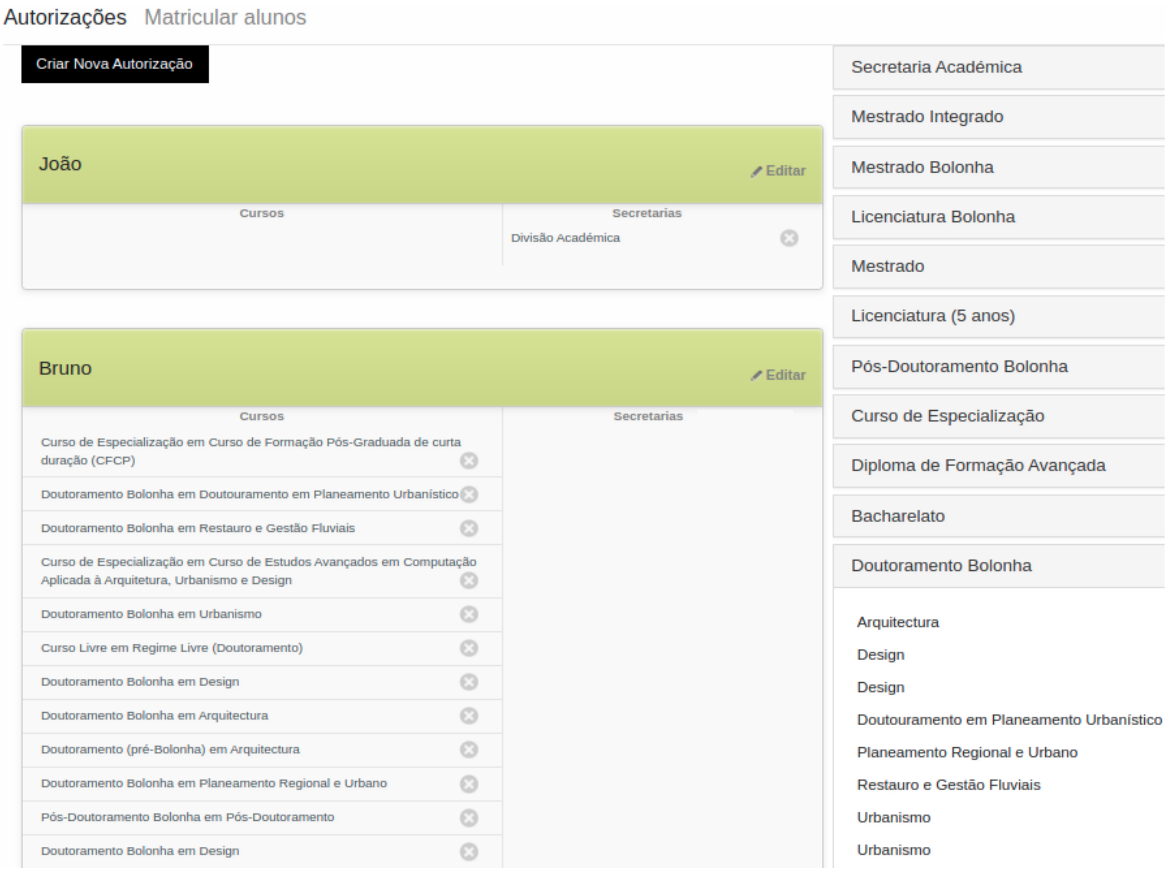

Figura 2.12: Grupos de permissões académicas

de acesso, são utilizados grupos com apenas um utilizador, como está representado na figura 2.13.

Os grupos de acesso são geridos através de uma funcionalidade na plataforma, acessível pelos administradores da plataforma ou utilizadores com autorização para gestão de autorizações.

A associação entre os grupos e os menus é efetuada através de expressões lógicas que permitem combinar grupos, através das operações de união, interceção, diferença e negação. Por exemplo, na figura 2.14, a expressão de acesso *"allStudents & (! candidate)"* define que apenas os utilizadores estudantes que não sejam candidatos têm acesso

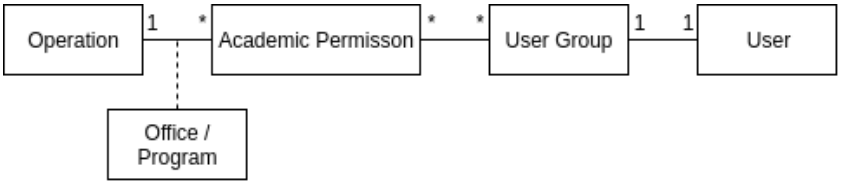

Figura 2.13: Modelo simplificado da permissão académica

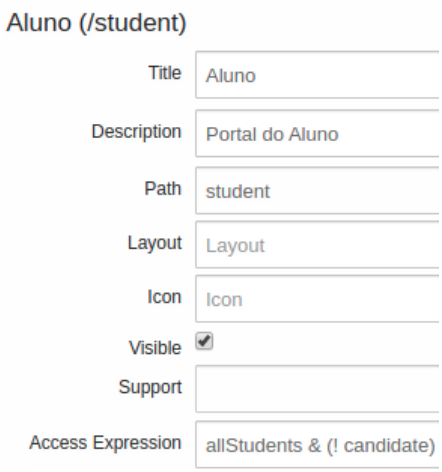

Figura 2.14: Interface do menu Aluno com expressão de acesso

ao menu "Aluno". As expressões utilizadas na composição de grupos são utilizadas para gerar novos grupos. No exemplo é gerado o grupo composto através da interceção do grupo *allStudents* com a negação do grupo *candidate*.

2.2.4 Desenho do controlo de acesso no Fenix

Nesta secção é apresentado o desenho do controlo de acesso no Fenix. Inicialmente é apresentado o diagrama simplificado de classes, passando de seguida para o diagrama de sequência de sistema.

Diagrama simplificado de classes do controlo de acesso do Fénix

A figura 2.15 apresenta o diagrama simplificado de classes que permitem concretizar o controlo de acesso no Fenix.

A ligação entre um menu e um utilizador é feita através de um grupo. Um utilizador só consegue aceder a funcionalidades através de um grupo, mesmo que esse grupo contenha apenas um utilizador.

Na figura 2.15, estão representados quatro tipos de grupos que têm como super classe a classe *Group*. Estes quatro grupos permitem uma implementação diferente da super classe *Group*, sem perder o funcionamento base da ligação original, com os utilizadores e os menus.

As classes *StudentGroup* e *ProfessorsGroup* correspondem a grupos geridos automaticamente pelo sistema. Estas classes determinam de forma automática os seus membros, de acordo com os atributos dos utilizadores.

A classe *DynamicGroup* corresponde aos grupos ad-hoc e apenas modifica a expressão e a forma como são adicionados e removidos utilizadores.

A classe *AcademicAuthorizationGroup* é utilizada para os grupos de permissões académicas. Tendo como base de funcionamento a super classe *Group*, acrescenta à sua implementação a possibilidade de restringir o âmbito de acesso do grupo, tal como explicado anteriormente na secção 2.2.3.

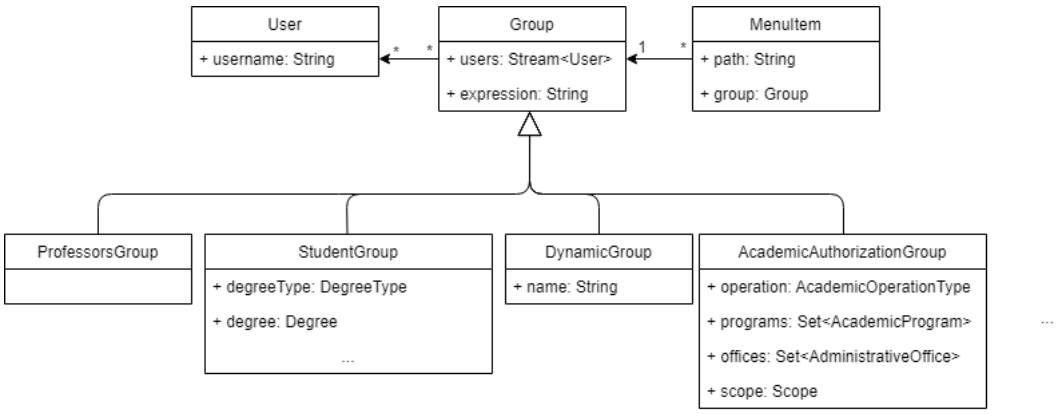

Figura 2.15: Diagrama simplificado de classes do controlo de acesso do Fénix

Diagrama de sequência de sistema do controlo de acesso

Conforme ilustrado na figura 2.16, quando um pedido chega ao servidor, este é intercetado pelo controlador de expedição para que seja determinado o menu associado, o *BennuPortalDispatcher*. Este controlador invoca o método *findFunctionalityWithPath* da classe *MenuContainer* para que seja identificado o *MenuItem* correspondente ao pedido. É verificado através do método *isItemAvailableForCurrentUser* se o utilizador tem acesso ao menu pretendido, esta verificação é efetuada por sua vez através do método *isMember* da classe *Group*, se o utilizador pertence ao mesmo.

Desta forma é possível fazer a verificação de acesso num único ponto do código evitando disseminar a verificação.

Adicionalmente existem exceções, em que há a necessidade de atribuir acesso a funcionalidades diferentes no mesmo ecrã a utilizadores com papéis diferentes. Por exemplo nas figuras 2.17 e 2.18, é possível observar a mesma interface em que na figura 2.17 o utilizador tem permissão para ver o processo do aluno e na figura 2.18 o utilizador não tem permissão para ver o processo do aluno, logo o botão não está disponível.

Quando existe esta necessidade, a verificação da autorização de acesso é efetuada na funcionalidade, conforme ilustrado na figura 2.19. Esta implementação permite que não se repita código e que haja coerência em relação aos dados que são apresentados na mesma interface.

Na figura 2.19 está representado um caso em que é verificado se o utilizador tem a autorização académica para gerir pautas. Se um utilizador pertencer ao grupo de per-

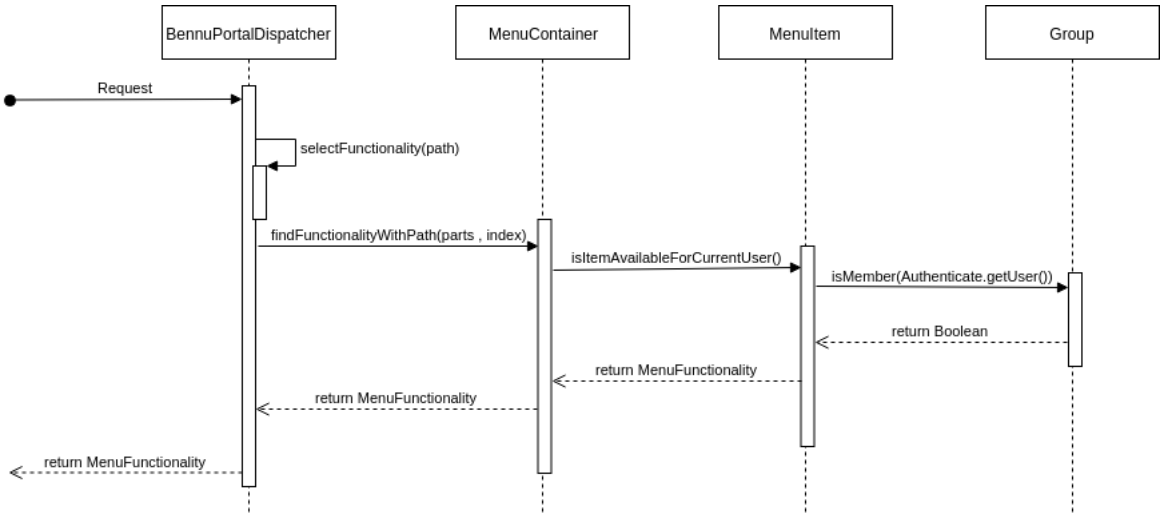

Figura 2.16: Diagrama de sequência de sistema do controlo de acesso

Ano Lectivo de Início	Data de Início	Curso	Estado Actual	Inscrições no Ano Corrente	Último Ano de Inscrição	
2017/2018	11-09-2017	Mestrado Integrado em Arquitetura (9257)	Matriculado	12	2018/2019	**Ver Processo »**

Figura 2.17: Interface com botão

Ano Lectivo de Início	Data de Início	Curso	Estado Actual	Inscrições no Ano Corrente	Último Ano de Inscrição	
2017/2018	11-09-2017	Mestrado Integrado em Arquitetura (9257)	Matriculado	12	2018/2019	

Figura 2.18: Interface sem botão

```
1146      private boolean generateGradeForEnrolmentEvaluation(final EnrolmentEvaluation evaluation, final HtmlTableRow row) {
1147          final Grade grade = evaluation.getGrade();
1149          final YearMonthDay availableDate = evaluation.getGradeAvailableDateYearMonthDay();
1150          final boolean isToShow =
1151                  !grade.isEmpty() && !evaluation.isTemporary() && availableDate != null && !availableDate.isAfter(new LocalDat
1152          final boolean markSheetAccess =
1153                  AcademicAuthorizationGroup.get(AcademicOperationType.MANAGE_MARKSHEETS).isMember(Authenticate.getUser());
1154
      ...
1169          final HtmlComponent component;
1170          if (markSheet == null || !markSheetAccess) {
1171              component = new HtmlText(text);
1172          } else {
1173              component = new HtmlLink();
1174
1175              ((HtmlLink) component).setText(text);
1176              ((HtmlLink) component).setModuleRelative(false);
1177              ((HtmlLink) component).setTarget("_blank");
1178              ((HtmlLink) component).setUrl(CompetenceCourseMarkSheetController.READ_URL + markSheet.getExternalId());
1179          }
1180          generateCellWithSpan(row, component, title, renderer.getGradeCellClass(), true);
1181          return isToShow;
1182      }
```

Figura 2.19: Autorizações em código, Classe StudentCurricularPlanLayout

missão académica para gerir pautas consegue aceder a uma funcionalidade disponibilizada por um link, caso contrário o utilizador apenas visualiza texto, sem qualquer link.

Esta implementação permite que a partir do mesmo menu a interface apresentada seja diferente consoante os grupos aos quais os utilizadores pertencem.

2.2.5 Interfaces de gestão do controlo de acesso no Fenix

Tal como referido na secção 2.2.3, o Fenix suporta três tipos de grupos. Os grupos que são geridos automaticamente pelo sistema não têm interface de gestão.

Na figura 2.20 é apresentada a interface para gestão dos grupos *ad-hoc*, referidos na secção 2.2.3, a qual permite adicionar membros, ver os membros atuais e eliminar membros de um grupo. Apesar de na imagem a designação dos grupos não incluir o prefixo cardinal (#), este é necessário na definição das expressões de controlo de acesso aos menus, conforme exemplificado na figura 2.21.

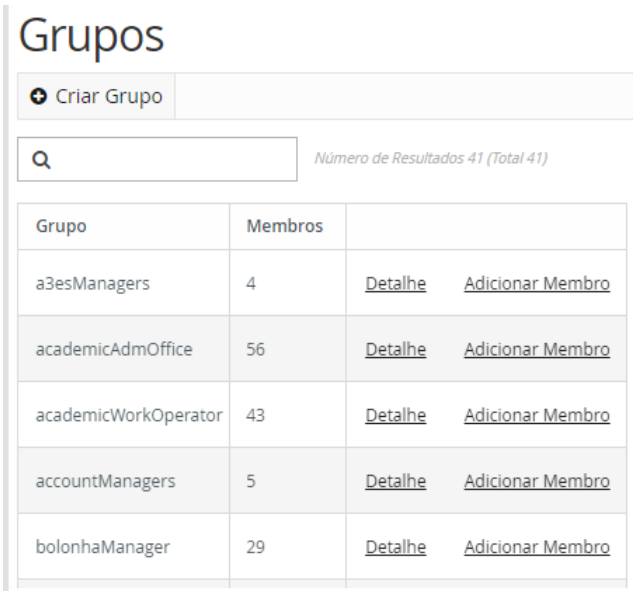

Figura 2.20: Interface gestão de grupos ad-hoc

A figura 2.21 ilustra a interface de gestão que permite associar grupos aos menus. Nesta interface, o campo *"Access Expression"* é utilizado para definir a expressão lógica de acesso. A imagem ilustra um exemplo de composição por união de grupos, com o grupo de permissão académica para matricular alunos e o grupo *ad-hoc* dos *managers*.

A interface 2.22, referente às permissões académicas, permite definir um nível mais granular de permissões. A interface está organizada por operações académicas e abrindo cada uma das operações académicas, obtemos a interface ilustrada na figura 2.23 que permite a remoção direta de utilizadores através do *link* remoção.

Recorrendo à funcionalidade de gestão da permissão, é apresentada uma outra interface, ilustrada na figura 2.24, que permite adicionar utilizadores e definir o âmbito da permissão (secretarias e cursos). A funcionalidade de gestão, apesar de bastante útil, não permite de forma ágil e eficiente a gestão de permissões académicas.

2.3 Sumário

Este capítulo está dividido em duas partes. Na primeira parte são apresentados os conceitos base de controlo de acesso e explicados mecanismos que servem como fundamento para o controlo de acesso aos sistemas.

Na segunda parte é apresentado o Fénix enquanto sistema integrado de gestão académica: arquitetura tecnológica, *frameworks* de desenvolvimento e lógica de funcionamento. É também referenciada, em maior detalhe, a implementação do controlo de acesso no sistema integrado de gestão académica Fenix.

No capítulo seguinte é descrito o trabalho realizado.

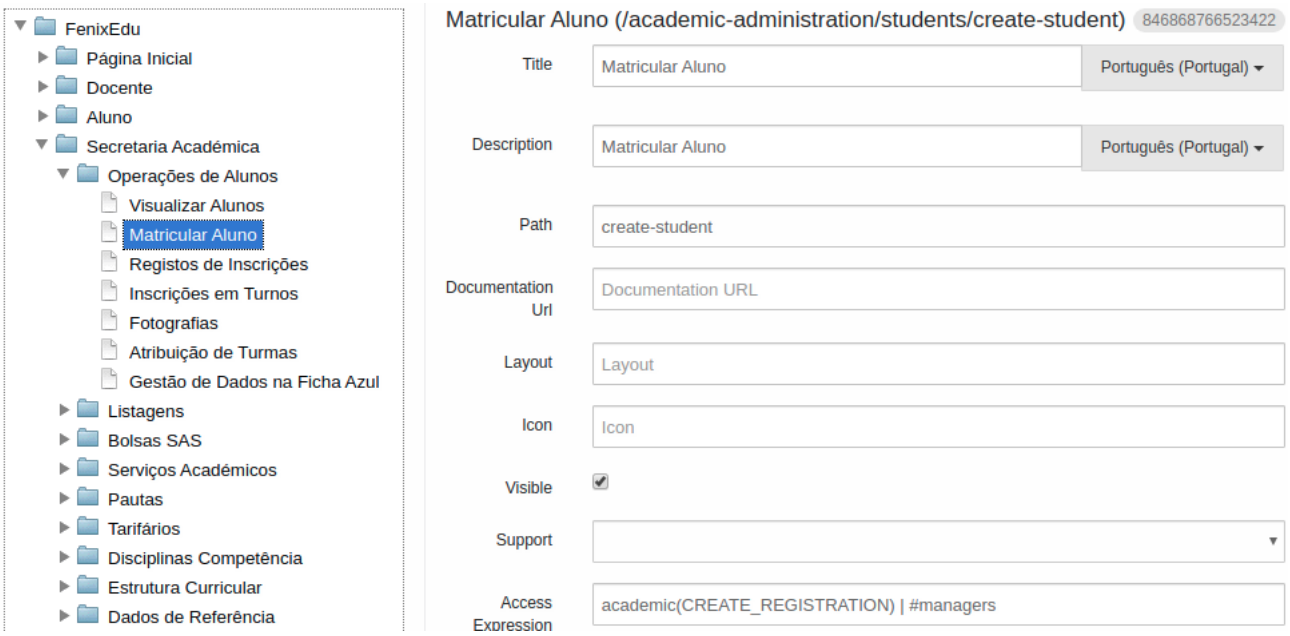

Figura 2.21: Interface configuração de grupos em menus

Figura 2.22: Interface de autorizações

Figura 2.23: Interface de autorizações com detalhes

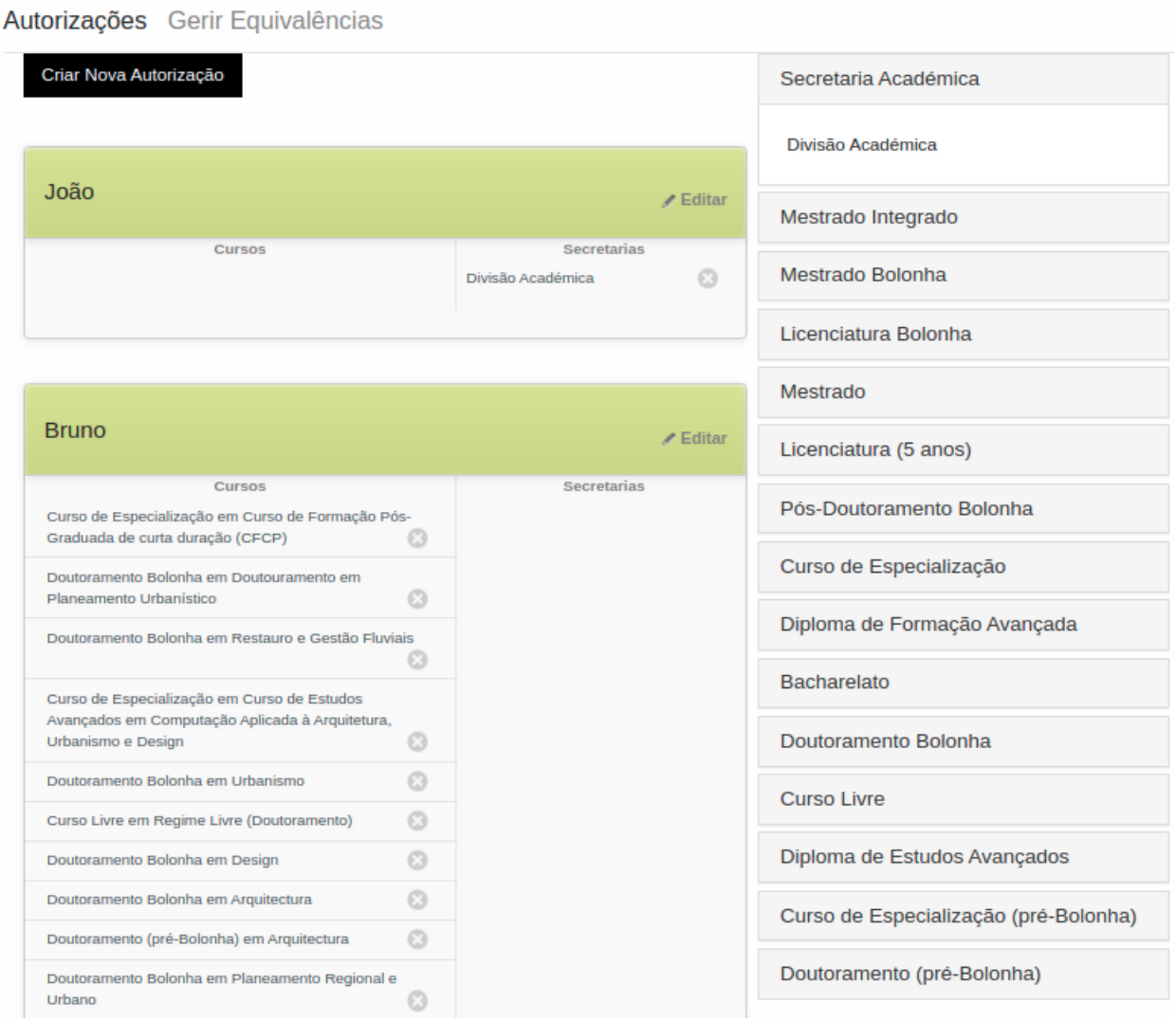

Figura 2.24: Interface de gestão de autorizações

Capítulo 3

Controlo de acesso baseado em perfis para o Fenix

Neste capítulo é proposto um modelo de controlo de acesso baseado em perfis para o sistema Fenix. O capítulo apresenta o trabalho desenvolvido desde a análise de requisitos à implementação da solução.

3.1 Análise de requisitos

A análise de requisitos foi feita com a colaboração da equipa responsável pelo suporte aplicacional do sistema Fenix dos serviços centrais da reitoria da Universidade de Lisboa, com membros da equipa de desenvolvimento do Fenix do Instituto Superior Técnico e com a empresa Quorum Born IT.

Adicionalmente, foram também analisados os requisitos do Regulamento Geral de Proteção de Dados [29].

O Regulamento Geral de Proteção de Dados, com entrada em vigor a 25 de Maio de 2018, regula o processamento de dados pessoais de indivíduos da União Europeia, por parte de indivíduos, companhias e organizações, quer seja de forma manual ou automática [29]. Este regulamento define um conjunto de regras que as entidades operantes na União Europeia têm de cumprir [29, Article 4].

O novo Regulamento Geral de Proteção de Dados levantou questões de uso e confidencialidade de dados pessoais, no sistema de gestão académica Fenix. São registados na ficha de indivíduo de alunos e docentes informações de carácter sensível, que necessitam de mecanismos que protejam o acesso ilícito e malicioso [29, Article 32]. Para além da proteção contra o acesso não autorizado à informação, é necessário responder a questões de auditoria e garantir que o acesso é única e exclusivamente feito por utilizadores autorizados, e para efeitos académicos [29, Article 25].

Figura 3.1: Utilizadores pertencentes a um grupo ad-hoc

3.1.1 Requisitos funcionais

Como resultado da análise foram identificados quatro tipos de requisitos funcionais, de seguida é descrito cada um deles.

1. Navegar entre conceitos de utilizadores, grupos e/ou permissões:
   (a) Dado um utilizador obter os seus grupos;
   (b) Dado um utilizador obter as suas permissões;
   (c) Dado um grupo obter os utilizadores;
   (d) Dado um grupo obter as suas permissões;
   (e) Dado uma permissão obter os utilizadores;
   (f) Dado uma permissão obter os grupos.

Atualmente, no sistema Fenix é possível obter apenas os utilizadores dos grupos *ad-hoc* e os utilizadores de uma determinada permissão académica. Por exemplo, na figura 3.1, estão apresentados os utilizadores que pertencem ao grupo *ad-hoc accountManagers*.

A figura 2.24 mostra que é possível obter os utilizadores com uma determinada permissão académica. Os restantes requisitos de navegação não são atualmente suportados pelo sistema fenix

2. Gerir permissões:

   (a) Permitir a cópia de permissões entre utilizadores;

   (b) Orientar a atribuição de permissões aos grupos.

   Atualmente o Fenix não permite fazer uma gestão de permissões orientada a grupos de utilizadores e tem as funcionalidades de gestão de grupos e permissões académicas espalhadas por várias interfaces. Por exemplo, para a substituição de funções de um utilizador por outro é necessário procurar o utilizador em todos os grupos e permissões académicas, em interfaces diferentes, e adicionar nas mesmas o utilizador novo pelo qual se pretende substituir.

   A atribuição de permissões académicas é feita utilizador a utilizador o que não é eficiente para atribuir permissões a um grande número de utilizadores. Desta forma existe a necessidade da atribuição de permissões a grupos.

3. Composição de grupos através de outros grupos:

   Com este requisito pretende-se a composição de grupos através de outros grupos para que não seja preciso repetir o processo de atribuição de permissões.

   Atualmente o Fenix permite fazer a composição de grupos, através das expressões de acesso, quando se define que grupo pode aceder a um determinado *menuItem*. No entanto esta composição é meramente gerida pela implementação do Fenix, não permitindo assim que os utilizadores façam a gestão destas composições.

   Estas composições apenas permitem a herança de utilizadores para o grupo composto e não a herança de permissões.

   No apêndice C está o guia de suporte às atribuições de permissões e grupos, utilizado pela equipa responsável pelo suporte aplicacional do sistema Fenix na atribuição de permissões e grupos aos utilizadores. Um exemplo em que a composição de grupos seria desejável, seria que o grupo "Super Utilizador de Secretaria" fosse composto através das permissões do grupo "Secretaria académica", ou seja o grupo "Super Utilizador de Secretaria" herdasse as permissões do grupo "Secretaria académica".

4. Permitir fazer a correspondência de grupos de utilizadores através da estrutura orgânica da instituição:

   A estrutura de grupos da plataforma deve estar associada a estrutura orgânica da instituição. Esta estrutura deve incluir o suporte hierárquico, para que seja mais fácil a gestão dos utilizadores, por exemplo, entrada e saída de utilizadores, alterações que ocorram na estrutura orgânica. Os grupos superiores devem herdar as permissões dos grupos inferiores.

3.1.2 Requisitos não funcionais

Durante a análise de requisitos forma também identificados requisitos não funcionais, descritos de seguida.

5. Manter um histórico de acessos no sistema [29, Article 30]:

   Este requisito obriga a que seja mantido um histórico de acessos. A plataforma Fenix já tem atualmente um sistema de *logging* que permite armazenar os acessos na plataforma, desta forma é possível rastrear as operações efetuadas.

6. Deve ser possível obter um relatório anual das atividades realizadas [29, Article 59]:

   Este requisito está muito relacionado com o anterior uma vez que apenas obriga a que o histórico de alterações em objectos seja disponibilizado para auditorias.

7. Simplificar a interface para a gestão de permissões:

   O utilizador tem de ser capaz de usar as funcionalidades sem precisar de formação sobres as mesmas. Um exemplo de má usabilidade é facto da funcionalidade de remoção de um utilizador de uma determinada permissão académica ser feita através de um clique na palavra "remover", como apresentado na figura 2.24. Esta palavra não tem qualquer tipo de destaque nem se parece com uma funcionalidade.

8. Desempenho:

   O sistema tem de ser capaz de responder de forma rápida às tarefas que o utilizador tem de desempenhar para a gestão de acesso.

3.1.3 Âmbito do trabalho

O trabalho desenvolvido para o sistema de controlo de acesso do Fenix responde a todos os requisitos funcionais e não funcionais identificados nesta secção, excepto o requisito funcional que consiste em fazer a correspondência de grupos de utilizadores com a da estrutura orgânica da instituição (descrito em maior detalhe no capítulo de trabalho futuro).

3.2 Desenho

Esta secção apresenta o desenho da solução final para que o controlo de acesso cumprisse os requisitos definidos.

Para permitir responder aos requisitos de navegação entre os conceitos e de composição de grupos, optou-se por aplicar o modelo de controlo de acesso baseado em perfis ao sistema Fenix. Ao contrário do comportamento base do grupo, o perfil suporta uma estrutura

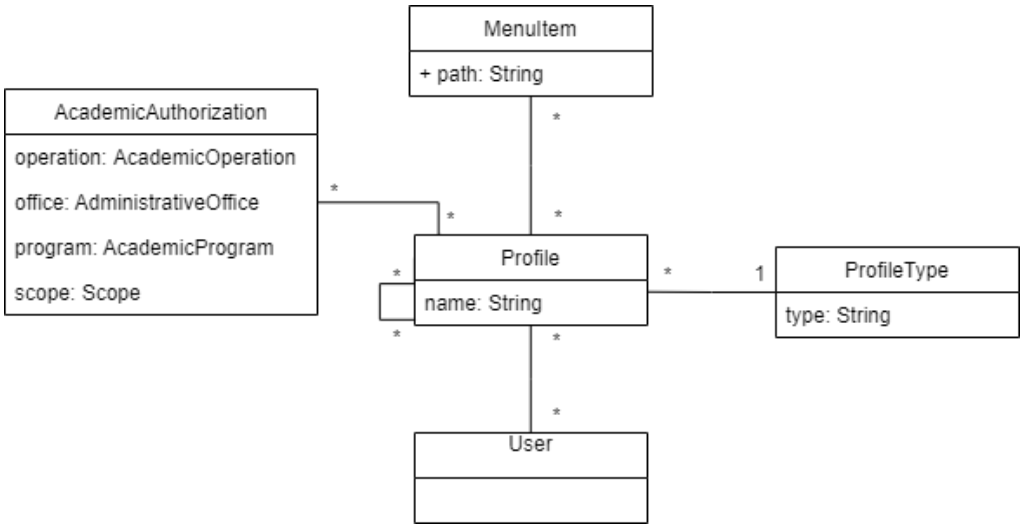

Figura 3.2: Diagrama de classes de controlo de acesso do sistema Fenix

hierárquica, permitindo assim que os perfis herdem as permissões dos perfis subordinados.

A figura 3.2 apresenta o diagrama de classes do controlo de acesso baseado em perfis para o Fenix. O perfil contém: nome, perfis pais que são perfis aos quais este é subordinado; perfis filhos que são perfis subordinados a este, um tipo de perfil e membros. O tipo de perfil permite definir se o perfil é de *Managers*, *Base* ou *General*. O perfil *Managers* é um perfil definido para os administradores da plataforma que gerem permissões, este perfil só é acessível por utilizadores administradores. O perfil *Base* é um perfil construído pelos utilizadores administradores, para serem usados como sub perfis pelos utilizadores não administradores que gerem as permissões em cada uma das escolas. O perfil *General* é um perfil construído pelos utilizadores não administradores que gerem as permissões em cada uma das escolas, que usam os perfis *Base* como filhos. Nestes perfis serão adicionados os utilizadores que devem ter acesso.

A permissão académica passa a estar ligada ao perfil e não ao grupo singular do utilizador, esta implementação facilita a gestão de permissões académicas porque desta forma todos os utilizadores de um determinado perfil têm a permissão.

A figura 3.3 representa o diagrama de sequência de sistema desde a receção do pedido até chegar ao perfil. Tal como referido anteriormente, quando um pedido chega ao servidor, este é intercetado pelo controlador de expedição para que seja determinado o menu associado, o *BennuPortalDispatcher*. Este controlador executa o método *"findFunctionalityWithPath"* da classe *MenuContainer* para que seja identificado o *MenuItem* correspondente ao pedido. É verificado através do método *"isItemAvailableForCurrentUser"* se o utilizador tem acesso ao menu pretendido, esta verificação é efetuada por sua vez através do método *"isMember"* da classe *Profile*, se o utilizador pertence a este grupo. A

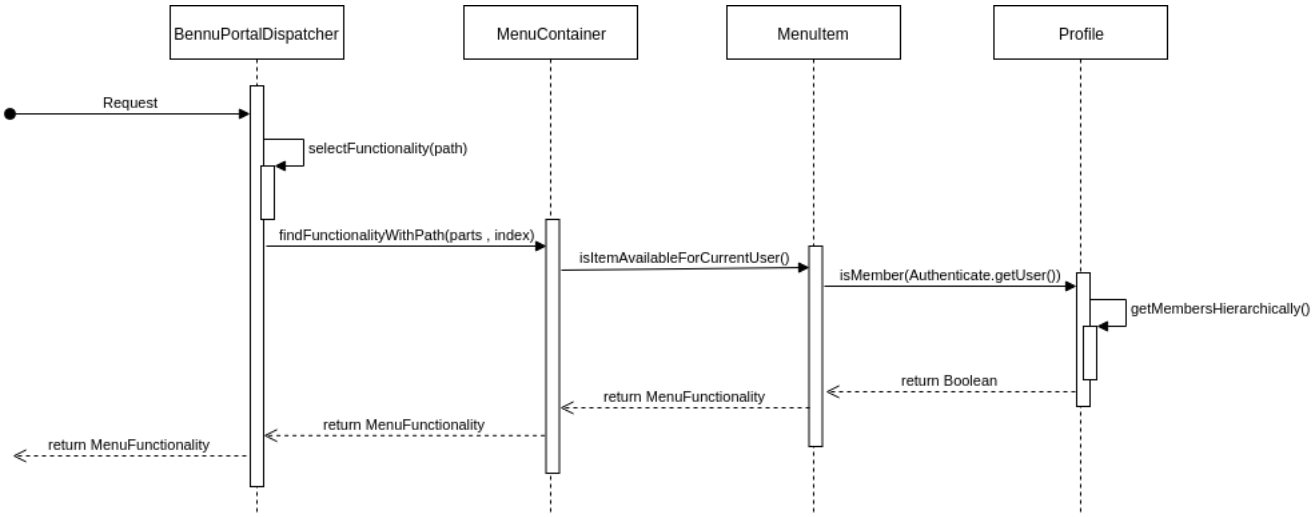

Figura 3.3: Diagrama de sequência de sistema Profile

classe *Profile* verifica hierarquicamente através do método *"getMembersHierarchically"* se o utilizador é membro do perfil e é devolvido o acesso ao menu.

3.3 Implementação

A solução foi desenvolvida em *Java*, utilizando a *framework* de desenvolvimento *web Spring* e segue o padrão *web MVC*, a *framework* Fenix e a *DML*. O *IDE* usado para o desenvolvimento de código foi o *Eclipse Java EE IDE for Web Developers* em *Linux* e para o versionamento foi usado o *GIT*. Para compilar o Fenix foi usada a ferramenta de compilação *Maven* e foi executada no servidor *web Tomcat*.

Inicialmente foi criado um módulo básico com as dependências necessárias, este módulo foi adicionado como dependência à *web app* principal do Fenix. É neste módulo que é implementado o novo modelo de controlo de acesso baseado em perfis.

Esta secção apresenta o modelo de classes, o desenvolvimento em código *Java*, a *DML* e a criação de interfaces.

3.3.1 Modelo de classes

A implementação da solução segue o modelo de classes apresentado na figura 3.4, foi criada a classe *Profile* que estende a já existente classe *Group* que define os comportamentos base dos grupos. A classe *Group* é imprescindível pois é uma classe abstrata e define o comportamento base dos grupos no Fenix.

Foi também criada a classe *ProfileType* para que seja atribuído um tipo de perfil ao perfil.

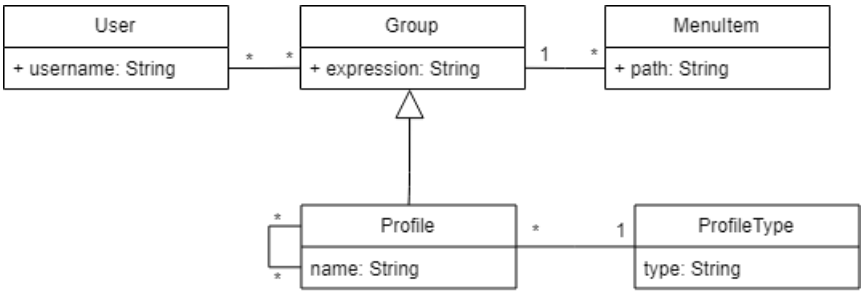

Figura 3.4: Modelo de classes dos perfis

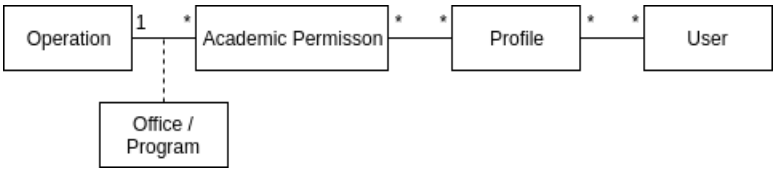

Figura 3.5: Modelo simplificado da permissão académica com ligação aos perfis

Com esta implementação, as permissões académicas, como ilustrado na figura 3.5, passam a estar ligadas ao perfil e não ao grupo singular do utilizador, ilustrado na figura 2.13. Desta forma cada objecto de uma determinada permissão académica pode estar associada a mais que um utilizador.

3.3.2 *DML*

A *DML* é onde se especifica os conceitos de domínio necessários para a persistência dos objectos. Uma vez que era necessário criar conceitos a nível de domínio, foram especificados estes conceitos através da *DML*, por sua vez estes conceitos são transformados em classes que são persistidas pela *Framework Fenix*. Por exemplo, na figura 3.6 está representada a especificação do tipo de perfil, do perfil, que estende a classe *PersistentGroup*, e da relação entre o tipo de perfil e o perfil.

Por uma questão de organização, estas classes pertence ao pacote *"groups"*. A classe *ProfileType* é referente ao tipo de perfil. A classe *PersistenteProfile* corresponde à persistência dos perfis, esta classe estende a classe *PersistenteGroup* para que herde o comportamento base da classe que é responsável pela persistência dos grupos.

A relação *ProfileType* corresponde à atribuição de um tipo de perfil a um perfil. É uma relação de 0 ou 1 tipos de perfil para muitos perfis.

```
 9 public class groups.ProfileType{
10     String type;
11 }
12
13 ...
14
15 public class groups.PersistentProfile extends .org.fenixedu.bennu.core.domain.groups.PersistentGroup {
16     public String name ;
17     protected String cod ;
18     protected DateTime created (REQUIRED);
19 }
20
21 relation ProfileType {
22     groups.PersistentProfileGroup playsRole group {
23         multiplicity *;
24     }
25     groups.ProfileType playsRole type {
26         multiplicity 0..1;
27     }
28 }
```

Figura 3.6: Excerto da especificação da *DML*

3.3.3 Código gerado pela *DML*

A especificação das duas classes e a relação entre as mesmas na *DML* permitem gerar duas classes. A classe perfil, figura 3.7, tem o comportamento base de se poder criar perfis e definir um tipo de perfil, entre outras funcionalidades. A classe do tipo de perfil, figura 3.8, permite criar tipos de perfil e obter todos os perfis associados a um tipo de perfil.

Os métodos das classes são gerados automaticamente consoante os atributos e as relações das classes. Por exemplo, na figura 3.7, o método *getName* foi gerado uma vez que a classe *PersistenteProfile* tem o atributo *Name*.

3.3.4 Extensão das classe base

Por sua vez, as classes geradas pela *DML* são estendidas para que se possa alterar o seu comportamento base.

Na figura 3.9 é apresentada a classe *PersistentProfile* que corresponde à persistência do perfil. Esta classe estende o comportamento da classe *PersistentProfile_Base*, sendo que altera a forma como o perfil é construído e a forma como são obtidos os membros, valida se não existem ciclos na árvore de hierarquia, apaga as relações de hierarquia antes de apagar o próprio perfil, entre outros.

Na figura 3.10 é apresentada a classe *ProfileType* que corresponde à persistência do perfil. Esta classe estende o comportamento da classe *ProfileType_Base*, sendo que altera a forma como o perfil é construído e implementa um método para que caso este exista obtêm esse tipo de perfil caso contrário cria-o.

```
4  @SuppressWarnings("all")
5  public abstract class PersistentProfile_Base extends org.fenixedu.bennu.core.domain.groups.PersistentGroup {
6
7  ...
8
9      // Constructors
10     protected  PersistentProfile_Base() {
13
14     // Getters and Setters
15
16     public java.lang.String getName() {
19
20     public void setName(java.lang.String name) {
23
24     protected java.lang.String getCod() {
27
28     protected void setCod(java.lang.String cod) {
31
32     protected org.joda.time.DateTime getCreated() {
35
36     protected void setCreated(org.joda.time.DateTime created) {
39
40     // Role Methods
41
42     ...
43
44     public org.fenixedu.accessControl.domain.groups.ProfileType getType() {
45         throw new UnsupportedOperationException("Not implemented in default code generator");
46     }
47
48     public void setType(org.fenixedu.accessControl.domain.groups.ProfileType type) {
49         throw new UnsupportedOperationException("Not implemented in default code generator");
50     }
51
52
53 }
```

Figura 3.7: Classe base gerada pela *DML* para a persistência de perfis

```
4  @SuppressWarnings("all")
5  public abstract class ProfileType_Base extends pt.ist.fenixframework.core.AbstractDomainObject {
6      // Static Slots
7
8      public static
   pt.ist.fenixframework.dml.runtime.DirectRelation<org.fenixedu.accessControl.domain.groups.ProfileType,org.fenixedu.accessControl.domain.groups.Persist
   entProfileGroup> getRelationProfileGroupType() {
11
12     public static
   pt.ist.fenixframework.dml.runtime.DirectRelation<org.fenixedu.accessControl.domain.groups.ProfileType,org.fenixedu.bennu.core.domain.Bennu>
   getRelationProfileTypes() {
15
16     // Constructors
17     protected  ProfileType_Base() {
18         super();
19     }
20
21     // Getters and Setters
22
23     public java.lang.String getType() {
26
27     public void setType(java.lang.String type) {
30
31     // Role Methods
32
33     public void addGroup(org.fenixedu.accessControl.domain.groups.PersistentProfile group) {
36
37     public void removeGroup(org.fenixedu.accessControl.domain.groups.PersistentProfile group) {
40
41     public java.util.Set<org.fenixedu.accessControl.domain.groups.PersistentProfile> getGroupSet() {
44
45     public org.fenixedu.bennu.core.domain.Bennu getBennu() {
48
49     public void setBennu(org.fenixedu.bennu.core.domain.Bennu bennu) {
52 }
```

Figura 3.8: Classe base gerada pela *DML* para a persistência de tipos de perfis

```
25 public class PersistentProfile extends PersistentProfile_Base {
26
27     protected PersistentProfile(String cod, PersistentGroup group) {
28         if (!cod.isEmpty()) {
29             this.setBennu(Bennu.getInstance());
30             this.setCod(cod);
31             this.setGroup(group);
32             this.setCreator(Authenticate.getUser());
33             this.setCreated(DateTime.now());
34         } else {
35             throw new Error("Profile name cannot be empty!");
36         }
37
38     }
39
41     public ProfileGroup toGroup() {
44
45     @Override
46     public Stream<User> getMembers() {
47         final Set<User> members = new HashSet<>();
48
49         members.addAll(this.getGroup().getMembers().collect(Collectors.toSet()));
50
51         getParentSet().forEach(Parent -> {
52             members.addAll(Parent.getMembers().collect(Collectors.toSet()));
53         });
54
55         return members.stream();
56     }
```

Figura 3.9: Classe PersistentProfile

```
10 public class ProfileType extends ProfileType_Base {
11
12     public ProfileType(String type) {
13         super();
14         this.setBennu(Bennu.getInstance());
15         this.setType(type);
16     }
17
18     @Atomic(mode = TxMode.WRITE)
19     public static ProfileType get(String type) {
20         final Optional<ProfileType> ptype = Bennu.getInstance().getProfileTypeSet().stream()
21                 .filter(profileType -> profileType.getType().equals(type)).findAny();
22         if (ptype.isPresent()) {
23             return ptype.get();
24         }
25
26         return new ProfileType(type);
27     }
28
29 }
```

Figura 3.10: Classe ProfileType

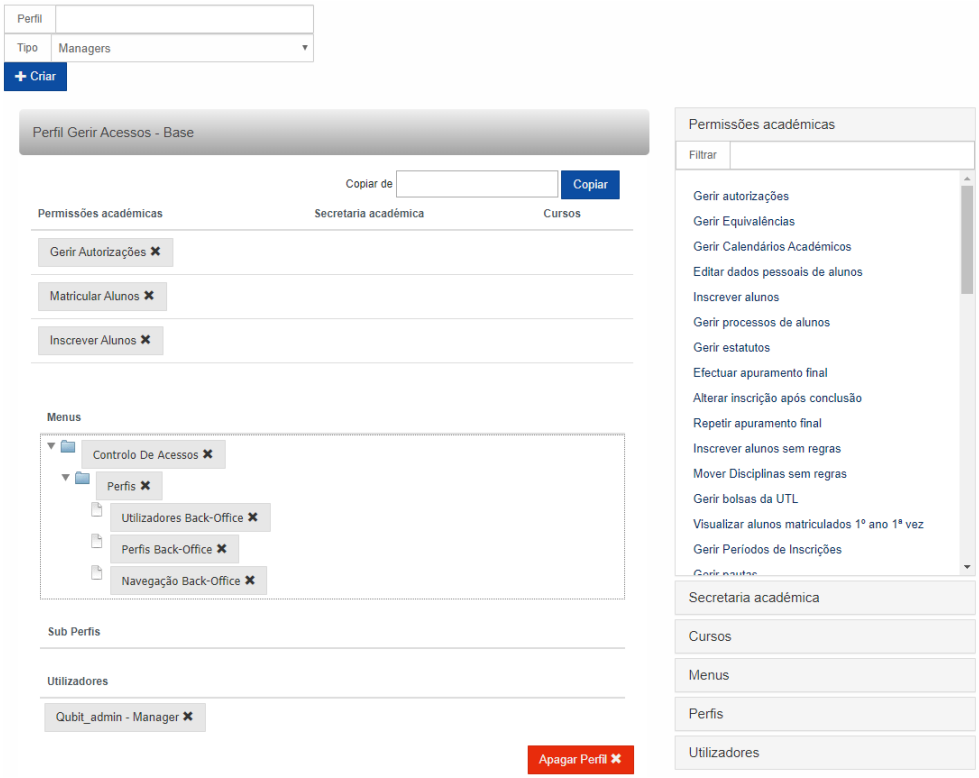

Figura 3.11: Interface de gestão de perfis

3.3.5 Interface

Juntamente com este novo conceito de perfil, foi criada uma nova interface, figura 3.11, que permite a gestão do controlo de acesso no sistema Fenix. A interface permite uma gestão orientada ao perfil, existindo uma segunda interface para a gestão de perfis orientada ao utilizador, figura 3.12.

A interface de gestão orientada ao perfil permite adicionar e remover utilizadores, permite adicionar e remover autorizações académicas, permite adicionar e remover os menus a que um determinado perfil dá acesso, assim como os sub perfis.

A interface de gestão orientada ao utilizador permite, adicionar e remover o utilizador a vários perfis, permite visualizar a que perfis este está adicionado e a que menus tem acesso. Outra funcionalidade desta interface é a cópia de perfis entre utilizadores.

3.4 Sumário

Este capítulo está dividido em três partes. Inicialmente é apresentada a análise de requisitos, explicando cada um deles em detalhe. De seguida, na segunda parte, é apresentado o desenho da solução.

Na terceira parte é descrita a implementação, descrevendo desde a construção do

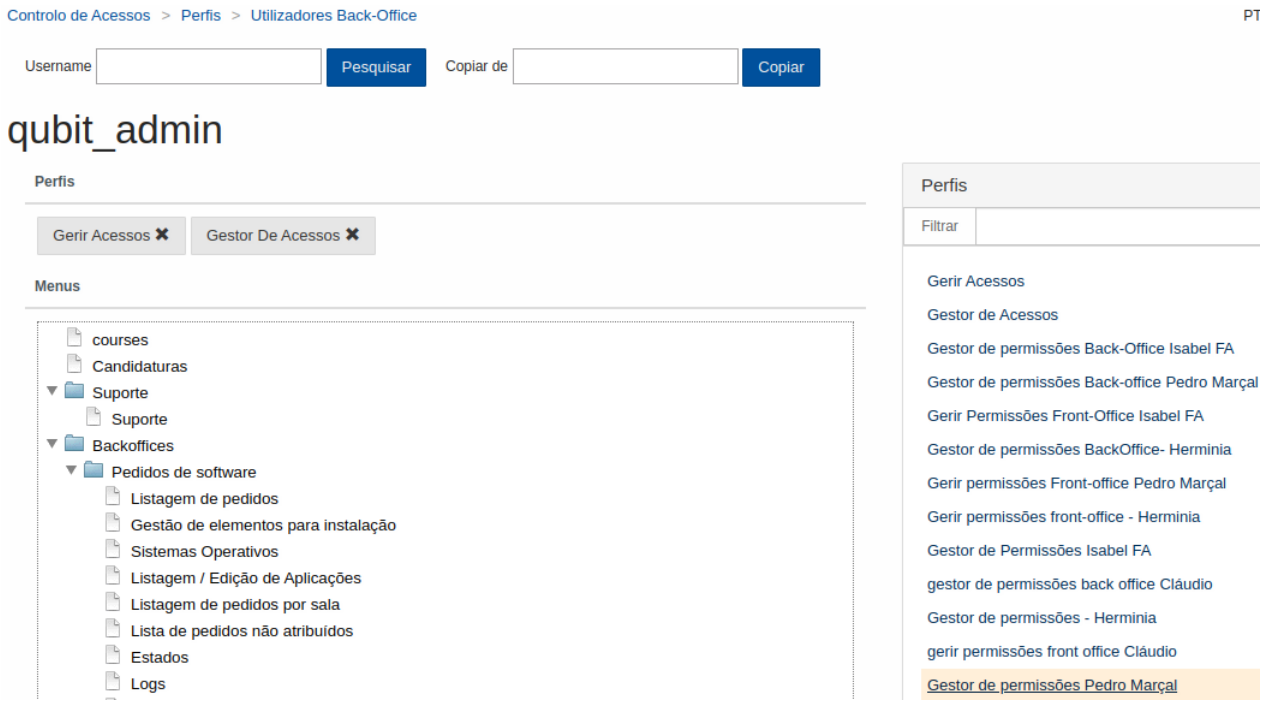

Figura 3.12: Interface de gestão de perfis orientada ao utilizador

módulo, a definição da *DML*, o código gerado pela mesma, a extensão das classes geradas e por fim a criação da interface.

No capítulo seguinte é feita a avaliação da implementação realizada.

Capítulo 4

Avaliação experimental

Este capítulo descreve o processo de avaliação experimental do módulo desenvolvido. Para avaliar a usabilidade da solução desenvolvida foi efetuado um questionário *System Usability Scale (SUS)*. Para que este questionário pudesse ser realizado foram feitos testes de usabilidade com recurso a cenários que simulam o uso normal do sistema. Estes cenários foram executados por utilizadores com funções de secretariado da plataforma e a sua execução é avaliada através de métricas, tais como o número de cliques, a duração da execução dos cenários, o número de erros cometidos e o número de vezes em que foi pedida ajuda.

4.1 System Usability Scale

Para avaliar a usabilidade do módulo desenvolvido foi efetuado um questionário *System Usability Scale*. Este questionário permite de um forma simples e rápida avaliar a usabilidade de um sistema numa escala de zero a cem. O questionário é respondido após o utilizador experimentar usar a plataforma sem qualquer tipo de apoio sobre o sistema [17].

O questionário consiste em 10 perguntas em que o utilizador tem que responder com uma escala de um a cinco, sendo que um corresponde a discordo bastante e cinco a concordo bastante. As perguntas estão feitas de forma a que o utilizador tenha que variar a escala consoante a pergunta para que não possa responder sempre o mesmo valor a todas as perguntas [17].

As perguntas ímpar quanto mais alto for a pontuação melhor é o resultado, as perguntas par quanto mais baixa é a pontuação melhor é o resultado.

Após os utilizadores responderem ao questionário é calculada uma pontuação que varia de zero a cem. Cada pergunta contribui com a posição da resposta, às respostas ímpar é subtraído 1 à posição da resposta, às respostas par é subtraindo a posição da resposta a cinco, soma-se todos os resultados e multiplica-se por dois e meio [17].

O apêndice A corresponde ao questionário realizado.

4.2 Testes com utilizadores

Para que fossem aplicados os questionários foram preparados cenários de utilização. Os cenários são compostos por tarefas de utilização normais do sistema, que correspondem à utilização do dia a dia dos utilizadores do sistema. Estes cenários tentam abranger o máximo de funcionalidades do sistema sem comprometer a complexidade da tarefa.

4.2.1 Caracterização dos utilizadores

Para os testes ao módulo de controlo de acesso, os utilizadores alvo são formados para o uso da plataforma e com funções de gestão de secretariado, alguns com conhecimento de gestão de acesso e outros sem qualquer tipo de experiência de gestão de acesso.

Tipicamente, estes utilizadores desempenham funções em secretarias académicas das escolas e nos serviços centrais da Universidade de Lisboa. São utilizadores formados para o uso da plataforma, devido à especificidade da mesma.

Foram contactadas algumas pessoas ligadas à gestão académica das várias escolas da Universidade de Lisboa, explicando que seriam utilizadores teste para o novo sistema de gestão de acesso. Das pessoas contactadas, dezanove mostraram-se disponíveis para realizar as tarefas e responder ao questionário.

4.2.2 Cenários

Os cenários permitem simular a utilização normal da plataforma para que se possa avaliar a execução de tarefas que levam à utilização das funcionalidades desenvolvidas.

Cada cenário pretende incluir uma ou mais funcionalidades do módulo de controlo de acessos de forma a abranger o máximo de funcionalidades possíveis do módulo.

Cenário 1

Este cenário pretende incluir funcionalidades de criação de perfil, adição de permissões académicas, menus e remoção de menus.

O utilizador deve:

1. Criar um perfil *Base* com o nome “Gestor de permissões”
2. Adicionar o menu “Controlo de Acessos”
3. Adicionar a permissão académica “Gerir autorizações”
4. Remover os menus com o nome de “Back-Office”

Cenário 2

Este cenário pretende incluir funcionalidades de criação de perfil, adição de sub-perfil e utilizador.

O utilizador deve:

5. Criar um perfil *General* "Gerir permissões Front-Office"
6. Adicionar o perfil "Gestor de permissões" como sub-perfil
7. Adicionar o utilizador "fc46571@alunos.fc.ul.pt - Daniel Filipe Mendes Pires" ao perfil "Gerir permissões Front-Office"

Cenário 3

Este cenário pretende incluir funcionalidades de criação de perfil, cópia de perfil. adição e remoção de menu.

O utilizador deve:

8. Criar um perfil *Base* com o nome "Gestor de permissões Back-Office"
9. Copiar a configuração do perfil "Gestor de permissões"
10. Retirar todos os menus
11. Adicionar os menus "back-office" da aba "Controlo de Acessos"

Cenário 4

Este cenário pretende incluir funcionalidades de pesquisa de utilizador, cópia de perfis entre utilizadores e remoção de perfis do utilizador.

O utilizador deve:

12. Pesquisar o utilizador "jplima@fc.ul.pt - José Pedro Galvão Lima"
13. Copiar os perfis do utilizador "fc46571@alunos.fc.ul.pt - Daniel Filipe Mendes Pires" para o utilizador "jplima@fc.ul.pt - José Pedro Galvão Lima"
14. Pesquisar o utilizador "fc46571@alunos.fc.ul.pt - Daniel Filipe Mendes Pires", de onde foi feito a cópia
15. Retirar o perfil "Gerir permissões Front-Office" a este utilizador

	Moda	Melhor resultado esperado
Pergunta 1	5	5
Pergunta 2	1	1
Pergunta 3	5	5
Pergunta 4	2	1
Pergunta 5	4	5
Pergunta 6	1	1
Pergunta 7	4	5
Pergunta 8	1	1
Pergunta 9	4	5
Pergunta 10	1	1

Tabela 4.1: Moda do resultado das perguntas inquérito SUS

4.2.3 Métodos de avaliação

Para avaliar a realização das tarefas serão contabilizadas métricas como o número de cliques, o tempo que demora a executar a tarefa, os erros cometidos e o número de vezes em que é pedida ajuda.

Estes fatores de avaliação permitem perceber a forma como os utilizadores se adaptaram à plataforma e evoluíram na utilização da mesma.

4.3 Análise dos resultados

Esta secção descreve os resultados obtidos dos questionários e das métricas da realização dos cenários.

4.3.1 Questionários

Através dos resultados obtidos do questionário *SUS* foi possível perceber que havia alguns aspetos a melhorar.

Fazendo uma análise pergunta a pergunta do questionário, tabela 4.1, e tendo em conta a moda de cada pergunta, foi possível observar que o novo modelo de controlo de acesso é uma ferramenta que os vários utilizadores gostariam de usar e que era algo simples de se usar, mas que há a necessidade de especificar melhor algumas funcionalidades e a formação de utilizadores para a utilização das mesmas.

Através da média e da moda obtidas dos resultados dos inquéritos, tabela 4.2 é possível comparar as mesmas numa escala de zero a cem e perceber que se encontram numa zona que é considerada bom [16]. Isto mostra que há ainda alguns pormenores para se retificarem, como por exemplo algumas questões de usabilidade que serão especificadas mais à frente, na secção 4.3.3.

Média	71,71052632
Mediana	75

Tabela 4.2: Média e Mediana do resultado do inquérito SUS

	Cliques	Tempo	Erros	Ajuda
Melhor	21	1 min 45 secs		
Intermédio	16	4 mins		
Pior	38	12 mins 30 secs	6	13

Tabela 4.3: Resultados Cenário 1

4.3.2 Cenários

Através da análise dos testes com recurso a cenários foi possível observar que existiam pequenos erros em relação à interface que podem ser melhorados. Para cada um dos cenários é apresentado o melhor resultado, um resultado intermédio e o pior resultado.

Cenário 1

No primeiro cenário, tabela 4.3, foi possível observar que os utilizadores de uma forma geral cometeram alguns erros devido ao primeiro impacto e à adaptação às novas interfaces. Todos os utilizadores foram capazes de acabar este cenário com sucesso.

Cenário 2

No segundo cenário, tabela 4.4, pode-se constatar que alguns utilizadores tiveram algumas dificuldades na funcionalidade de arrasto sendo que conseguiram corrigir e completar a tarefa com sucesso sem ajuda.

Cenário 3

O terceiro cenário, tabela 4.5, pretendia que se removesse todos os menus do perfil que podia ser efetuado de duas formas. Metade dos utilizadores efetuou da forma mais eficiente, removendo o menu agrupado enquanto que os restantes removeram item a item. Neste cenário os utilizadores precisaram de ajuda para remover os menus pois não era

	Cliques	Tempo	Erros	Ajuda
Melhor	13	1 min 19 secs		
Intermédio	9	4 mins	1	2
Pior	13	13 mins 13 secs	2	

Tabela 4.4: Resultados Cenário 2

	Cliques	Tempo	Erros	Ajuda
Melhor	16	1 min 30 secs		
Intermédio	21	2 mins 16 secs	1	
Pior	9	4 mins		2

Tabela 4.5: Resultados Cenário 3

	Cliques	Tempo	Erros	Ajuda
Melhor	10	1 min		
Intermédio	13	3 mins e 47 secs	1	
Pior	6	5 mins		1

Tabela 4.6: Resultados Cenário 4

explícito que se removia com o duplo clique, estando à espera que fosse através de um botão de apagar.

Cenário 4

O quarto cenário, tabela 4.6, era um cenário bastante simples, os utilizadores conseguiram efetuar as tarefas rapidamente sem dificuldades, sendo que apenas dois utilizadores voltaram a pesquisar o utilizar em vez de usar a caixa própria para efetuar a cópia.

4.3.3 Alterações

Tendo em conta os resultados apresentados foi possível perceber que havia algumas alterações a nível de interfaces que permitiriam melhorar o uso das mesmas. Assim, as alterações efetuadas passaram por acrescentar filtros de pesquisa nas barras laterais, alterar a forma como se apagam os menus dos perfis, alterar o texto do botão de apagar o perfil para se tornar mais claro que o botão apaga o perfil e mostrar uma ajuda para onde arrastar os itens das barras laterais.

4.4 Sumário

Este capítulo foi divido em duas partes. Inicialmente é descrito o questionário, os cenários e a caracterização dos utilizadores.

Na segunda parte é feita a análise dos dados. Com a avaliação experimental foi possível observar que os utilizadores de forma geral adaptaram-se bem à plataforma, mas houve alguns erros que puderam ser solucionados com as alterações feitas.

No capítulo seguinte é descrito o trabalho futuro.

Capítulo 5

Conclusões e trabalho futuro

5.1 Conclusões

O Fenix é uma plataforma para gestão académica que se encontra implementada em dezasseis escolas da Universidade de Lisboa. Tem inúmeras funcionalidades com acesso restrito em que é necessário a existência de um sistema de controlo de acesso. No entanto, a gestão de permissões através do sistema de controlo de acesso atual não permite a gestão de uma forma ágil, necessitando de demasiadas operações o que torna a gestão complexa. Esta dificuldade ocorre devido à forma como foi criada a interface e como está implementado o controlo de acesso.

Após o estudo sobre os conceitos base do controlo de acesso, sobre o Fenix e feita a análise de requisitos, foi criado um modelo de controlo de acesso baseado em perfis para o Fenix. A sua concretização mudou a forma como o controlo de acessos está implementado e criou uma interface nova que responde aos requisitos identificados.

Para avaliar a solução implementada, foram criados testes com utilizadores, recorrendo a cenários de utilização do sistema de controlo de acesso e foi aplicado um questionário. Os resultados dos testes com cenários mostraram que os utilizadores conseguiram realizar as tarefas com sucesso mas que precisaram alguma ajuda. Através dos resultados dos inquéritos foi possível concluir que o sistema de controlo de acesso é algo que os utilizadores desejariam usar para a gestão de permissões mas que precisariam de algum tipo de formação para a sua utilização.

O trabalho desenvolvido permitiu evoluir a forma como estavam a ser geridas as permissões de acesso na plataforma. Como os resultados da avaliação puderam provar, a gestão de permissões tornou-se num processo ágil e rico em funcionalidades que permitiu dar liberdade aos gestores de permissões de cada escola, uma vez que cumpriu os requisitos definidos, como a navegação entre conceitos, novas funcionalidades de gestão de permissões como a cópia, a composição de perfis através de outros perfis e a simplificação de interfaces.

Como trabalho futuro prevê-se a entrada em produção e a ligação com a estrutura

orgânica.

5.2 Trabalho futuro

O trabalho futuro inclui o que é necessário fazer de futuro para que o novo modelo de controlo de acessos baseado em perfis para o Fenix responda a todos os requisitos e entre em produção.

5.2.1 Entrada em produção

Esta secção descreve as tarefas necessárias a serem feitas para a entrada em produção do módulo de controlo de acesso.

1. Definir os perfis base

Tendo em conta que o controlo de acesso desenvolvido permite a criação de perfis com base noutros perfis e que as escolas não têm permissões para criar perfis *base*, há a necessidade de preparar o sistema para que seja possível disponibilizar um conjunto de perfis *base* a serem usados pelas escolas para a construção de novos perfis *general*.

Para efetuar a identificação das permissões mais usadas em conjunto foi extraído do Fenix um *dataset* de permissões usadas pelos utilizadores. O *dataset* foi extraído através de um *script* corrido na plataforma Fenix.

Este *dataset* foi posteriormente usado por algoritmos de análise de dados para que fosse possível extrair a relação entre si. Os algoritmos usados foram o *Apriori* e o *Association Rules* em *Python*. Estes algoritmos permitem explorar conjuntos frequentes de itens que ocorrem em simultâneo num determinado conjunto de dados.

Na figura 5.1 é apresentado uma parte do gráfico com o resultado dos algoritmos de análise de dados, onde é possível observar que, por exemplo, 77.19% dos utilizadores que têm a permissão para efetuar listagens de alunos (*STUDENT_LISTINGS*) e ver o currículo do aluno (*VIEW_FULL_STUDENT_CURRICULUM*) também têm a permissão de gerir pagamentos de alunos (*MANAGE_STUDENT_PAYMENTS_AD*).

Com as relações identificadas entre as várias permissões académicas e grupos, através dos algoritmos, é possível agrupar por perfis as permissões académicas e grupos usados em conjunto. Estes perfis vão servir como base de construção de perfis do tipo *base*, que por sua vez são usados para a construção dos perfis *general*.

2. Atribuir utilizadores aos perfis base

Para que os utilizadores possam usufruir das permissões e que haja continuidade no funcionamento do sistema é necessário fazer a atribuição de utilizadores aos perfis.

De forma a efetuar esta atribuição são verificadas as permissões de todos os utilizadores e colocados esses utilizadores nos perfis onde estas permissões se encontram.

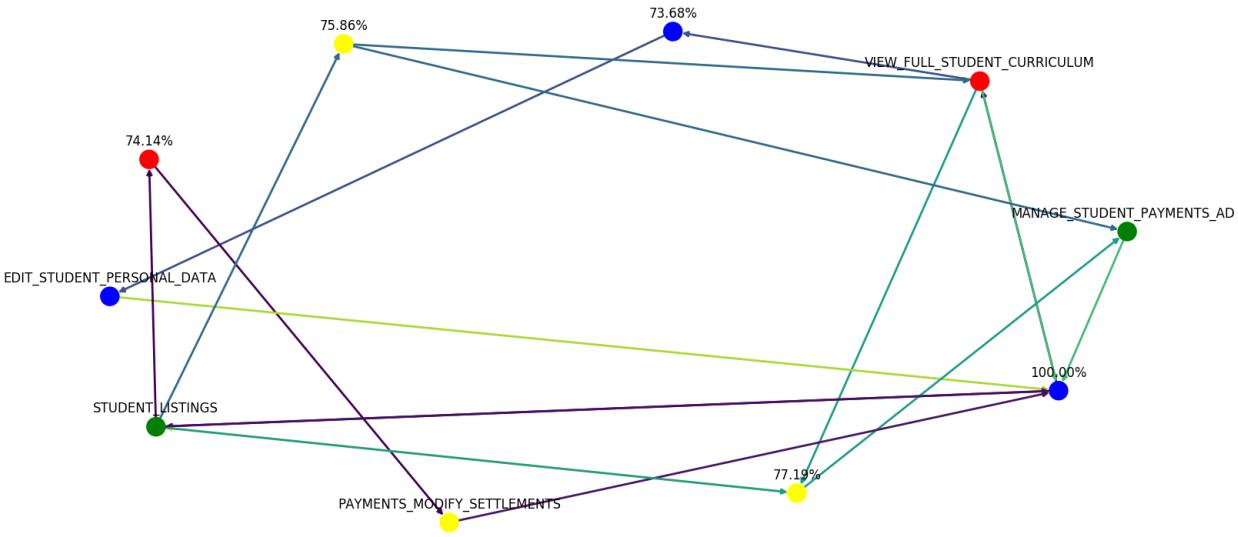

Figura 5.1: Gráfico com relações de permissões

3. Documentação e formação

De acordo com a política da equipa de suporte dos serviços centrais foi criado um documento que explica o funcionamento do módulo de controlo de acesso e as suas funcionalidades, como por exemplo a criação, remoção e edição de perfis.

Adicionalmente, está prevista a realização de acções de formação para os gestores de permissões das escolas.

5.2.2 Estrutura organizacional

Nenhum dos modelos de controlo de acesso apresentados anteriormente satisfaz um modelo que se adapte à estrutura da organização. Desde muito que nos é introduzido o conceito de estrutura orgânica, por exemplo, uma organização é composta por departamentos, os departamentos podem ser compostos por núcleos que por sua vez têm várias pessoas.

Uma estrutura organizacional estática não permite a flexibilidade de introduzir mais elementos na estrutura sem ter que alterar a sua implementação, como é o caso das diferentes orgânicas da escolas. Por exemplo, na figura 5.2 está representada a estrutura orgânica da Faculdade de Ciências e na figura 5.3 a estrutura orgânica da Faculdade de Letras, é possível observar diferenças nas duas estruturas.

Adicionalmente é necessário suportar uma estrutura hierárquica múltipla, ou seja, em que uma unidade da orgânica tem mais do que um pai. Este tipo de hierarquia é bastante útil, por exemplo, quando uma cadeira é lecionada em mais que um curso.

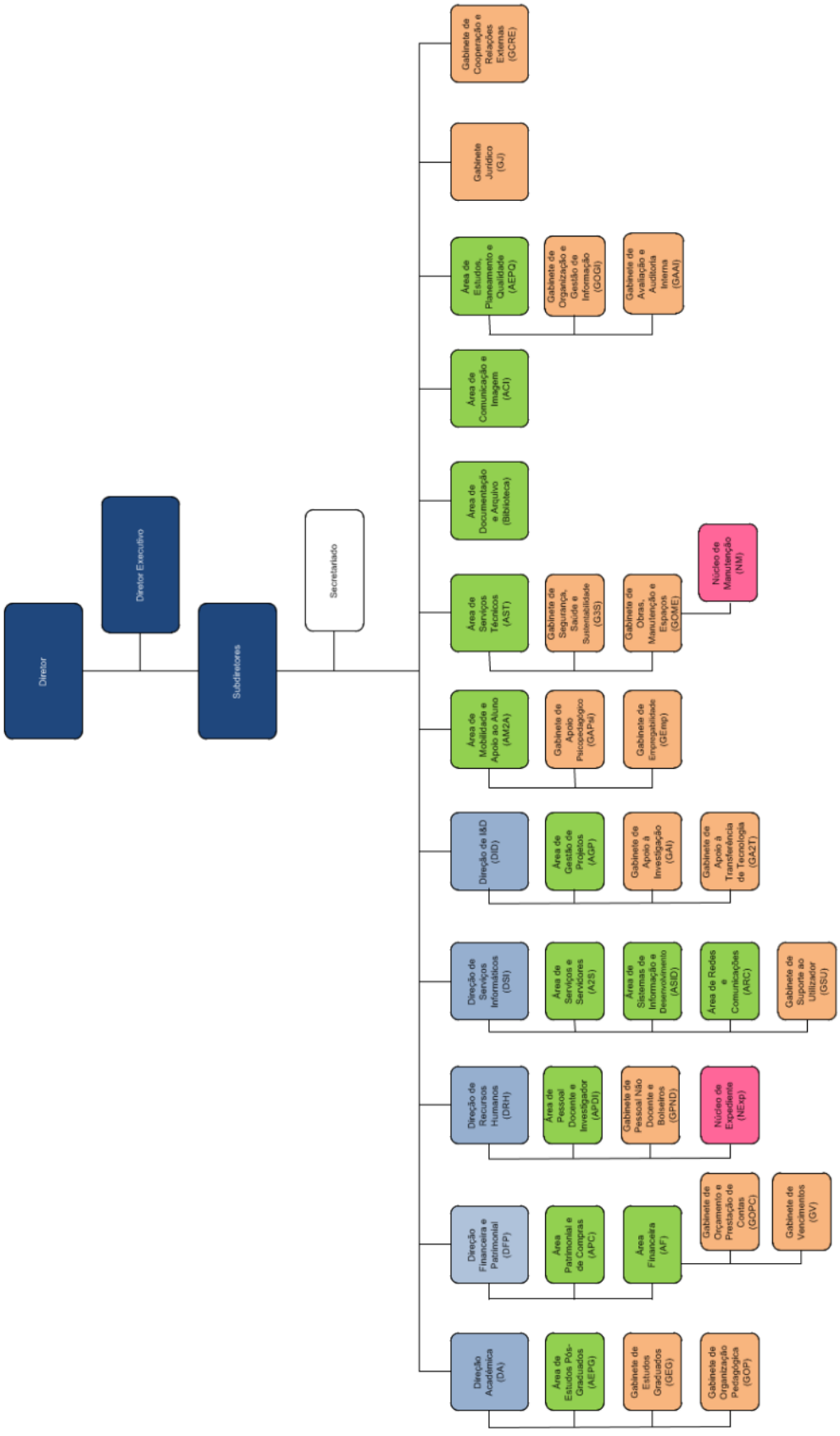

Figura 5.2: Estrutura orgânica da Faculdade de Ciências

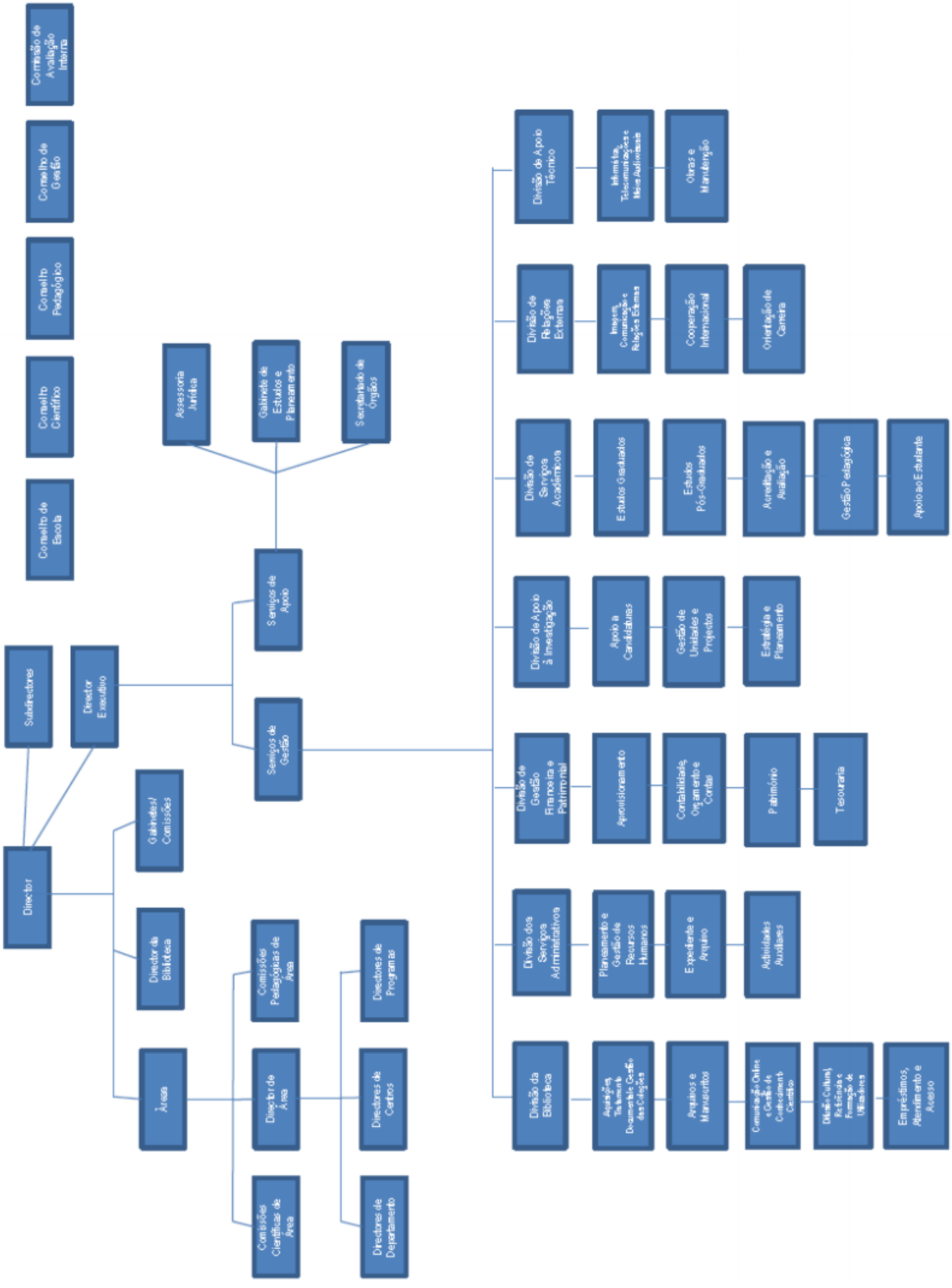

Figura 5.3: Estrutura orgânica da Faculdade de Letras

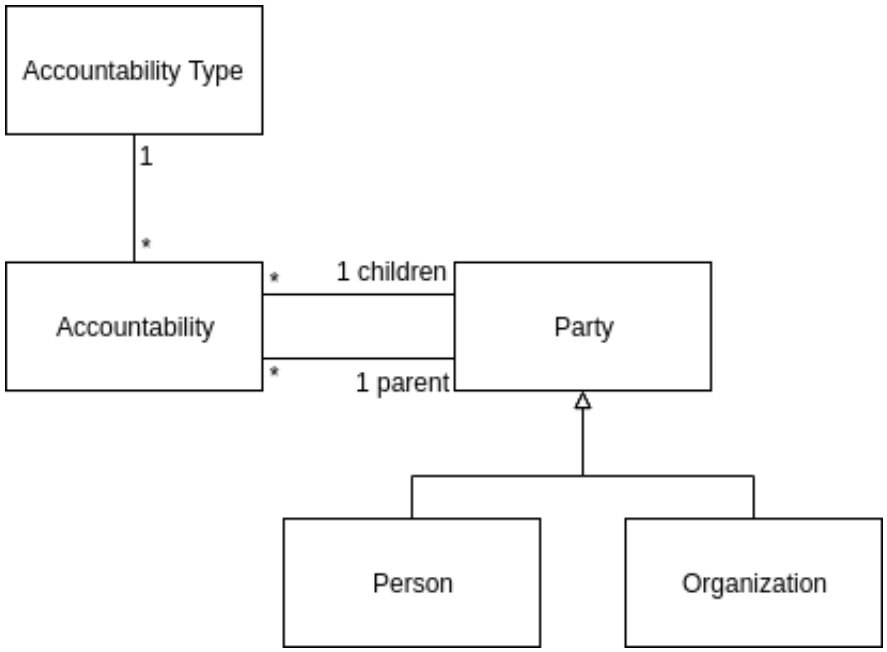

Figura 5.4: Estrutura organizacional com *Accountability*

Na figura 5.4 está apresentado um modelo de estrutura organizacional com o conceito *Accountability* que permite responder as requisitos identificados.

O conceito de *Accountability* permite criar tipos de relações entre as *Party*. A *Party* é uma abstração dos elementos que compõem a organização.

O *Accountability* permite o crescimento da organização de forma hierárquica definindo os vários tipos diferentes de relações entre as *Party* [21]. Cada uma destas relações tem um tipo que representa a natureza da ligação e desta forma é possível gerir qualquer número de relações organizacionais [21].

De forma a criar a ligação da estrutura orgânica ao controlo de acesso baseado em perfis, a classe Perfil iria estender a classe *Party*, para que fosse possível criar a ligação entre as unidades orgânicas e o perfil através do conceito de *Accountability*, conform ilustrado na figura 5.5.

Por exemplo, na figura 5.6, está exemplificado um caso da ligação da classe Perfil com a classe *Organization* através do *Accountability*. O "Presidente DI"tem uma relação do tipo "ACRole"com o perfil "Presidente DI".

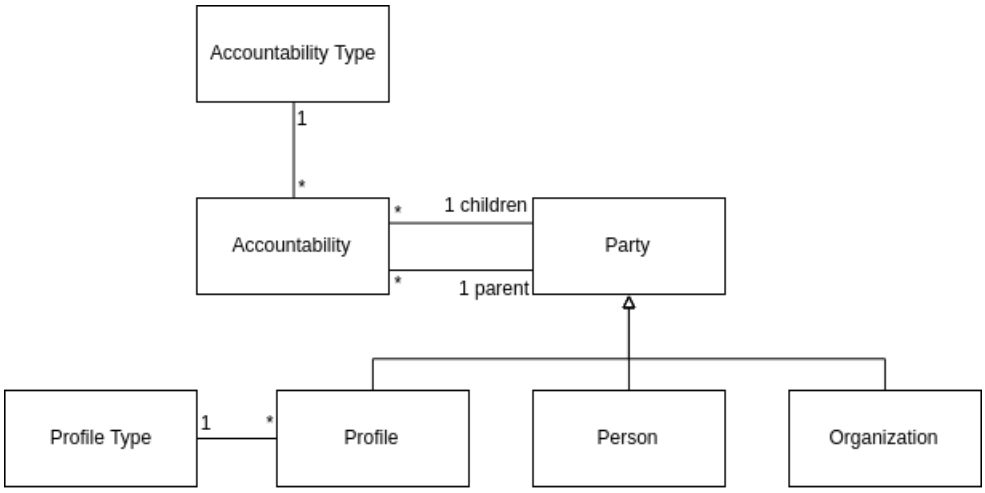

Figura 5.5: Perfis com *Accountability*

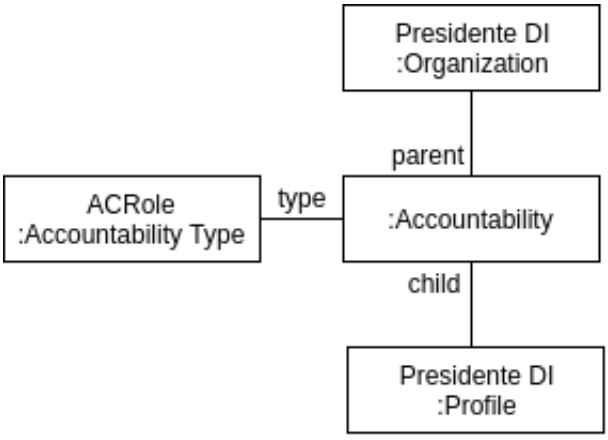

Figura 5.6: Exemplo de perfis com *Accountability*

Bibliografia

[1] Angularjs — superheroic javascript mvw framework. https://angularjs.org/. (Acedido em 01/2019).

[2] Fenix domain browser. https://fenix-ashes.ist.utl.pt/fdb/. (Acedido em 10/2018).

[3] Fenixedu confluence. https://confluence.fenixedu.org/display/FENIXEDU/Welcome. (Acedido em 10/2018).

[4] Fenixedu/bennu: Framework for developing java web applications based on the fenix framework. https://github.com/FenixEdu/bennu. (Acedido em 01/2019).

[5] Fenixedu/fenixedu-academic: Fenixedu academic is open source student information system. https://github.com/FenixEdu/fenixedu-academic. (Acedido em 01/2019).

[6] Fenixedu™. https://fenixedu.org/. (Acedido em 10/2018).

[7] Fénix framework. https://fenix-framework.github.io/. (Acedido em 01/2019).

[8] Fénix framework. https://fenix-framework.github.io/DML.html. (Acedido em 01/2019).

[9] Maven – welcome to apache maven. http://maven.apache.org/. (Acedido em 01/2019).

[10] Mysql. https://www.mysql.com/. (Acedido em 01/2019).

[11] Oracle technology network for java developers — oracle technology network — oracle. https://www.oracle.com/technetwork/java/index.html. (Acedido em 01/2019).

[12] Spring framework overview. https://docs.spring.io/spring-framework/docs/5.1.6.RELEASE/

spring-framework-reference/overview.html#overview. (Acedido em 01/2019).

[13] Welcome - bennu - fenixedu confluence. https://confluence.fenixedu.org/display/BENNU/Welcome. (Acedido em 01/2019).

[14] Welcome to the apache struts project. https://struts.apache.org/. (Acedido em 01/2019).

[15] Ryan Ausanka-Crues. Methods for access control: Advances and limitations.

[16] Aaron Bangor, Philip Kortum, and James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123, 2009.

[17] John Brooke. Sus - a quick and dirty usability scale.

[18] João Cachopo and António Rito-Silva. Versioned boxes as the basis for memory transactions. *Science of Computer Programming*, 63(2):172–185, 2006.

[19] João Manuel Pinheiro Cachopo. Development of rich domain models with atomic actions, 2007.

[20] David F Ferraiolo, Ravi Sandhu, Serban Gavrila, D Richard Kuhn, and Ramaswamy Chandramouli. Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, 2001.

[21] Martin Fowler. Organization structures.

[22] Avraham Leff and James T Rayfield. Web-application development using the model/view/controller design pattern. In *Proceedings fifth ieee international enterprise distributed object computing conference*, pages 118–127. IEEE, 2001.

[23] Elizabeth J O'Neil. Object/relational mapping 2008: hibernate and the entity data model (edm). In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1351–1356. ACM, 2008.

[24] Pierangela Samarati and Sabrina Capitani de Vimercati. Access control: Policies, models, and mechanisms. In *International School on Foundations of Security Analysis and Design*, pages 137–196. Springer, 2000.

[25] Ravi Sandhu. Roles versus groups. In *Proceedings of the first ACM Workshop on Role-based access control*, page 7. ACM, 1996.

[26] Ravi S Sandhu. Role-based access control. In *Advances in computers*, volume 46, pages 237–286. Elsevier, 1998.

[27] William Stallings and Lawrie Brown. *Computer Security: Principles and Practice*. Pearson, 2015. Third Edition.

[28] Instituto Superior Técnico. The fenixedu project: an open-source academic information platform. 2011.

[29] European Union. Regulamento geral sobre a proteção de dados. *Official Journal of the European Union*, 59, May 2016.

Apêndice A

Usabilidade do sistema

*Obrigatório

1. **Gostaria de usar este sistema** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo bastante	◯	◯	◯	◯	◯	Concordo bastante

2. **Achei o sistema desnecessariamente complexo** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo bastante	◯	◯	◯	◯	◯	Concordo bastante

3. **Achei que o sistema foi fácil de utilizar** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo bastante	◯	◯	◯	◯	◯	Concordo bastante

4. **Penso que iria precisar do suporte de alguém especializado para poder usar este sistema** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo bastante	◯	◯	◯	◯	◯	Concordo bastante

5. **Achei que as várias funcionalidades do sistema estavam bem integradas** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo bastante	◯	◯	◯	◯	◯	Concordo bastante

6. **Achei que havia demasiada inconsistência neste sistema** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo bastante	◯	◯	◯	◯	◯	Concordo bastante

7. **Imagino que a maioria das pessoas iria aprender a usar este sistema muito rapidamente** *
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo bastante	◯	◯	◯	◯	◯	Concordo bastante

8. **Achei o sistema muito incómodo de utilizar** *
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo bastante	◯	◯	◯	◯	◯	Concordo bastante

9. **Senti-me muito confiante ao utilizar o sistema** *
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo bastante	◯	◯	◯	◯	◯	Concordo bastante

10. **Precisaria de aprender muitas coisas antes de me poder habituar a este sistema** *
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo bastante	◯	◯	◯	◯	◯	Concordo bastante

Apêndice B

Resultados Questionário																				
Questão	**Utilizador 1**	**Utilizador 2**	**Utilizador 3**	**Utilizador 4**	**Utilizador 5**	**Utilizador 6**	**Utilizador 7**	**Utilizador 8**	**Utilizador 9**	**Utilizador 10**	**Utilizador 11**	**Utilizador 12**	**Utilizador 13**	**Utilizador 14**	**Utilizador 15**	**Utilizador 16**	**Utilizador 17**	**Utilizador 18**	**Utilizador 19**	**Moda**
1	5	4	4	5	4	3	5	5	3	3	5	5	4	5	5	5	4	5	5	5
2	1	2	1	2	4	3	1	3	3	3	1	1	5	1	1	2	2	1	4	1
3	5	5	4	3	2	3	5	4	4	2	5	5	2	5	5	4	4	4	4	5
4	2	3	1	4	5	5	2	4	3	4	1	2	5	1	4	2	2	2	3	2
5	4	4	5	4	3	3	5	4	4	3	4	5	3	5	4	4	4	3	4	4
6	1	2	1	2	2	3	1	2	4	2	1	1	2	1	3	1	3	2	1	1
7	4	4	4	4	2	1	4	3	4	2	4	5	3	5	4	4	2	4	4	4
8	1	1	1	1	2	3	2	2	2	3	1	1	4	1	1	2	2	5	2	1
9	5	4	4	3	4	2	4	4	4	3	4	5	3	5	4	3	4	5	4	4
10	1	1	1	2	4	3	4	2	2	4	1	1	5	1	2	2	1	4	1	1
Resultado	92,5	80	90	70	45	37,5	82,5	67,5	62,5	42,5	92,5	97,5	35	100	77,5	77,5	70	67,5	75	90
Média	71,71052632																			
Mediana	75																			

Apêndice C

Financeiro / Tesouraria

- grupo a atribuir
    - employees
- permissões respectivas
    - Gerir pagamentos de alunos **
    - Gerir pagamentos de Alunos (Avançado) **
    - Tesouraria: Lançamento de notas de dívida **
    - Tesouraria: Lançamento de pagamentos **
    - Gerir processos de alunos **Inscrever alunos **
    - Efectuar listagens de alunos **
    - Ver o currículo do aluno **

Leitura (Secretaria Académica / Tesouraria)

- grupo a atribuir
    - employees
- permissões respectivas
    - Gerir processos de alunos **
    - Efectuar listagens de alunos **
    - Ver o currículo do aluno **
    - Gerir pagamentos de alunos (perfil de leitura da tesouraria) + **

+Caso não se pretenda dar acesso à tesouraria não incluir esta permissão

Reitoria (acesso às instâncias de cada escola)

- grupo a atribuir
    - academicAdmOffice
    - employees
    - raidesOperator
    - socialServicesOperator (permissões bolsas SAS)
- permissões respectivas
    - Matricular alunos **
    - Editar dados pessoais de alunos **
    - Efectuar apuramento final **
    - Gerir Equivalências **
    - Gerir actividades extra-curriculares
    - Gerir pautas **
    - Gerir processos de alunos **
    - Gerir estatutos
    - Processar serviços académicos **
    - Inscrever alunos **
    - Efectuar listagens de alunos **
    - Ver o currículo do aluno **
    - Gerir unidades externas
    - Gerir processos de candidatura **
    - Gerir pagamentos de alunos **
    - Gerir pagamentos de Alunos (Avançado) **

    - Tesouraria: Lançamento de pagamentos **
    - Tesouraria: Lançamento de notas de dívida **

Algumas destas permissões (como o Matricular alunos) permitem a um utilizador gerir todos os cursos (nesta situação o campo Secretaria deverá ser preenchido com a Divisão Académica) ou caso se pretenda restringir o acesso ao nível do curso, será necessário especificar quais a gerir no campo Cursos.

Super Utilizador de Secretaria

- permissões respectivas (perfil secretaria académica):
    - Alterar inscrição após conclusão **
    - Repetir apuramento final **
    - Inscrever alunos sem regras ** (APENAS PARA PERFIL DE CHEFE DE SECRETARIA) *
    - Mover Disciplinas sem regras ** (APENAS PARA PERFIL DE CHEFE DE SECRETARIA) *

* NÃO ATRIBUIR ESTA PERMISSÃO A OUTROS UTILIZADORES SEM PERCEBER EXACTAMENTE A MOTIVAÇÃO, CASO CONTRÁRIO, PODEM ESTAR A SER OCULTADOS PROBLEMAS DE DADOS E/OU CONFIGURAÇÃO

Gestão Curricular

- grupo a atribuir
    - bolonhaManager
    - scientificCouncil
- permissões respectivas
    - Gerir a estrutura de ensino **

Inquéritos

- grupo a atribuir
    - surveyManager

Gestão de Serviço Docente

- grupo a atribuir
    - teacherServiceManagers

Gestão de Espaços

- grupo a atribuir
    - spaceSuperUsers

Planeamento e Recursos

- grupo a atribuir
    - resourceAllocationManager
    - spaceSuperUsers
    - operator
- permissões respectivas
    - Gerir Períodos de Inscrições **
    - Gerir autorizações de docência
    - Gerir atribuição de serviço de docência
    - Gerir Calendários Académicos
    - Gerir Disciplinas de Execução **
    - Gerir Disciplinas de Execução (Avançado)

Formação avançada (menu qubIT)

- grupo a atribuir
    - academicWorkOperator

Utilizadores (menu qubIT)

- grupo a atribuir
    - academicWorkOperator

Gestão de candidaturas

- grupo a atribuir
    - candidacyManager

Gerir Mapeamentos RAIDES

- grupo a atribuir
    - raidesManager

Configuração de Prescrições

- grupo a atribuir
    - schooladmin

Nota: grupo de administrador da escola (permitirá efetuar a gestão dos utilizadores no suporte; Atribuição de Turmas; Alocação a Turnos; CMS; Candidaturas 1º ano 1ª vez; Sincronização LDAP; Atribuição Automática de Turmas)

** requer associar à permissão a Divisão académica